

National Information Exchange Model Naming and Design Rules

Draft Version 1.2 August 7, 2007

Editors:

Webb Roberts, Georgia Tech Research Institute
Susan Liebeskind, Georgia Tech Research Institute
Mark Kindl, Georgia Tech Research Institute

Abstract:

This document specifies the data model, XML components, and XML data for use with the National Information Exchange Model (NIEM) version 2.0.

Status:

This document is a draft specification for NIEM-conformant XML components. It represents the design that has evolved from the collaborative work of the NIEM Business and Technical Architecture Committees (NBAC and NTAC) and their predecessors.

This specification is a product of the NIEM Program Management Office (PMO), but has NOT been officially approved by either the PMO or the NIEM governance committees (NBAC and NTAC). The PMO has recommended that this document be published for public review at the same time the PMO, NBAC, and NTAC are reviewing it.

Send comments on this specification via email to
`niem-comments@lists.gatech.edu`.

24	Table of Contents	
25	1. Introduction	4
26	1.1. Scope	4
27	1.2. Audience	5
28	1.3. Document Conventions	5
29	1.4. Terminology	6
30	1.5. Document Organization	7
31	2. The NIEM Conceptual Model	9
32	2.1. NIEM Data Objects	9
33	2.2. NIEM Data Assertions	10
34	2.3. NIEM Data Model Explicit Not Implicit	10
35	2.4. NIEM Data Model Implementation in XML Schema	11
36	3. Guiding Principles	13
37	3.1. Specification Guidelines	13
38	3.2. XML Schema Design Guidelines	13
39	3.3. Modeling Design Guidelines	16
40	3.4. Implementation Guidelines	16
41	4. Relation to Standards	19
42	4.1. XML 1.0	19
43	4.2. XML Namespaces	19
44	4.3. XML Schema	19
45	4.4. ISO 11179, Part 4	19
46	4.5. ISO 11179, Part 5	21
47	5. XML Schema Design Rules	23
48	5.1. Restrictions on XML Schema Constructs	23
49	5.2. <code>xsd:schema</code> Document Element	30
50	5.3. Namespace Imports	31
51	5.4. Annotations	34
52	5.5. Type Definitions	36
53	5.6. Additional Definitions And Declarations	38
54	6. Modeling Rules	40
55	6.1. <code>xsd:schema</code> Document Element Restrictions	40
56	6.2. Annotations	41
57	6.3. Complex Type Definitions	48
58	6.4. Component Usage	54
59	6.5. NIEM Structural Facilities	55
60	6.6. Using External Schemas	58
61	6.7. Container Elements	61
62	7. XML Instance Rules	63
63	7.1. Instance Validation	63
64	7.2. Instance Meaning	63
65	7.3. Component Representation	64
66	7.4. Component Ordering	65
67	7.5. Instance Metadata	66

68	8. Naming Rules	68
69	8.1. Extension of XSD Namespace Simple Types	68
70	8.2. Usage of English	68
71	8.3. Characters in Names	68
72	8.4. Character Case	69
73	8.5. Use of Acronyms and Abbreviations	69
74	8.6. Word Forms	70
75	8.7. Name Generation	71
76	8.8. Object Class Term	71
77	8.9. Property Term	72
78	8.10. Qualifier Terms	72
79	8.11. Representation Term	72
80	8.12. NIEM Type Names	75
81	8.13. NIEM Property Names	76
82	Appendix A. NIEM Overview	79
83	Appendix B. NIEM Design Principles	81
84	Appendix C. NIEM Rules	83
85	Appendix D. Name Syntax for Special Components	100
86	Appendix E. Representation Terms	101
87	Appendix F. Documentation Standard Opening Phrases	103
88	Appendix G. NIEM Core Abbreviations	104
89	Appendix H. Supporting Schemas	105
90	Appendix I. References	121
91	Appendix J. Glossary	123
92	Appendix K. Notices	126
93		

94

1. Introduction

95 This Naming and Design Rules (NDR) document specifies schemas for use with the
96 National Information Exchange Model (NIEM). The NIEM is an information sharing
97 framework based on the World Wide Web Consortium (W3C) eXtensible Markup
98 Language (XML) Schema standard. In February 2005, the U.S. Departments of Justice
99 (DoJ) and Homeland Security (DHS) signed a cooperative agreement to jointly develop
100 the NIEM by leveraging and expanding the Global Justice XML Data Model (GJXDM) into
101 multiple domains. The NIEM is a result of a combined government and industry effort to
102 improve information interoperability and exchange within the U.S. at federal, state, tribal,
103 and local levels of government.

104 NIEM specifies a set of reusable information components for defining standard
105 information exchange messages, transactions, and documents on a large scale: across
106 multiple communities of interest and lines of business. These reusable components are
107 rendered in XML schemas as type, element and attribute definitions that comply with the
108 W3C XML Schema specification. The resulting reference schemas are available to
109 government practitioners and developers at <http://niem.gov/>.

110 The W3C XML Schema standard enables information interoperability and sharing by
111 providing a common language for describing data precisely. The constructs it defines are
112 basic metadata building blocks – baseline data types and structural components. Users
113 employ these building blocks to describe their own domain-oriented data semantics and
114 structures. Rules that profile allowable XML Schema constructs and describe how to use
115 them help ensure that those components are consistent and reusable.

116 This document specifies principles and enforceable rules for NIEM data components and
117 schemas. Schemas and components that obey the rules set forth here are considered to
118 be **NIEM-conformant**.

1.1. Scope

120 This document is a specification for NIEM 2.0. It is not intended to specify beyond the
121 NIEM 2.0 release. The document covers the following issues in depth:

- 122 • The underlying NIEM data model
- 123 • Guiding principles behind the design of NIEM
- 124 • Rules for using XML Schema constructs in NIEM
- 125 • Rules for modeling and structuring NIEM-conformant schemas
- 126 • Rules for creating NIEM-conformant instances
- 127 • Rules for naming NIEM components

128 This document does NOT address the following:

- 129 • A formal definition of the NIEM data model.
130 Such a definition would focus on the Resource Definition Framework (RDF) and
131 concepts not strictly required for interoperability. This document instead focuses
132 on definition of schemas that work with the data model, to ensure translatability
133 and interoperability.
- 134 • A detailed discussion of NIEM architecture and schema versioning.
135 Such rules will be addressed in **[ARCH]**.
- 136 • The artifacts of the NIEM information exchange process.
137 The artifacts of the NIEM information exchange process are discussed in **[IEPD]**.

138 This document is intended as a technical specification. It is not intended to be a tutorial or
139 a user guide. A brief NIEM Overview is provided in Appendix A.

140 **1.2. Audience**

141 This document is targeted at government practitioners and developers who employ XML
142 for information exchange and interoperability. Such information exchanges may be
143 between organizations or within organizations. The NIEM reference schemas provide
144 system implementers much content on which to build specific exchanges. However,
145 there is a need for extended and additional content. The purpose of this document is to
146 define the rules for such new content so that it will be consistent with the NIEM reference
147 schemas. These rules are intended to establish and, more importantly, enforce a degree
148 of standardization on a national level.

149 **1.3. Document Conventions**

150 This document uses formatting and syntactic conventions to clarify meaning and avoid
151 ambiguity.

152 **1.3.1. Document References**

153 This document relies on references to many outside documents. Such references are
154 noted by bold, bracketed inline terms. For example, a reference to RFC 2119 is shown
155 as **[RFC2119]**. All reference documents are recorded in Appendix I, References.

156 **1.3.2. Normative and Informative Content**

157 This document includes a variety of content. Some content is normative (binding and
158 enforceable in implementations), while other content is informative (explanatory, but not
159 part of the NIEM specification). In general, the informative material appears as
160 supporting text and specific rationales for the normative material.

161 Conventions used within this document include:

162 **[Definition: <term>]**

163 A formal definition of a term associated with NIEM.

164 Definitions are normative.

165 **[Principle <number>]**

166 A guiding principle for NIEM.

167 The principles represent the requirements, concepts, and goals that have helped
168 shape the NIEM. Principles are informative, not normative, but act as the basis
169 on which the rules are defined.

170 Principles are accompanied by a short discussion section that justifies the
171 application of the principle to NIEM design.

172 **[Rule <section>-<number>]**

173 An enforceable rule for NIEM.

174 Rules state specific requirements on artifacts, such as schemas and instances.
175 Most rules apply to conformant schemas while others apply to instances. The
176 rules are normative.

177 Rules are stated using both XML InfoSet terminology (elements and attributes)
178 and XML Schema terminology (schema components). The choice of terminology
179 is driven by which standard best expresses the rule. Certain concepts are more
180 clearly expressed using XML InfoSet information items, others using the XML

181 Schema data model, and still others are best expressed using a combination of
182 terminology drawn from both standards.

183 Rules have rationales which justify the need for the rule. For clarity, there may be
184 multiple rules which have the same rationale.

185 Rules and supporting text may use Extended Backus-Naur Form (EBNF)
186 notation as defined by **[XML]**.

187 Rules are numbered according to the section in which they appear, and the order
188 in which they appear within that section. For example, **[Rule 4-1]** is the first rule
189 in Section 4. Rule identifiers that are deleted or re-categorized will not be reused
190 until a major release milestone is reached, at which point all identifiers may be
191 reset.

192 **1.3.3. Formatting**

193 In addition to special formatting for definitions, principles and rules, this document uses
194 consistent formatting to identify NIEM components.

195 *Courier*: All words appearing in *Courier* font are values, objects, keywords, or literal
196 XML text.

197 *Italics*: All words appearing in *italics*, when not titles or used for emphasis, are special
198 terms with definitions appearing in this document.

199 Keywords: Keywords reflect concepts or constructs expressed in the language of their
200 source standard. Keywords have been given an identifying prefix to reflect their source.
201 The following prefixes are used:

- 202 • `xsd:` identifies keywords from the W3C XML Schema Definition Language
203 specification.
- 204 • `xsi:` identifies keywords from the W3C XML Schema's XML Schema Instance
205 specification.
- 206 • `structures:` identifies keywords from the NIEM structures namespace.
- 207 • `appinfo:` identifies keywords the NIEM appinfo namespace.

208 Throughout the document, fragments of XML schema or XML instances are used to
209 clarify a principle, or rule. These fragments are specially formatted in *Courier* font, and
210 appear in text boxes. An example of such a fragment would appear like this:

```
211 <xsd:complexType name="PersonType">  
212   ...  
213 </xsd:complexType>
```

214 **1.4. Terminology**

215 This document uses standard terminology to explain the principles and rules that
216 describe NIEM.

217 **1.4.1. RFC 2119 Terminology**

218 Within normative content (rules and definitions), the key words MUST, MUST NOT,
219 REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY,
220 and OPTIONAL in this document are to be interpreted as described in **[RFC2119]**

221 .

222 1.4.2. XML Information Set Terminology

223 This document uses the concepts of element information items (“element”) and attribute
224 information items (“attribute”) and their associated properties as defined by **[XMLInfoSet]**
225 with clarifications as discussed below. Note that in the clarification that follows, the
226 abstract property names appear in square brackets relative to the information item to
227 which they belong. For example, “Element[parent]” discusses the abstract property
228 “parent” of the element information item.

- 229 • parent of an element (Element[parent])

230 child of an element (Element[children])

231 Note that the InfoSet properties “Element[parent]” and “Element[children]”
232 correspond to a direct, immediate relationship with an element. Children of an
233 element, and their children, and so on, will be collectively referred to as
234 “descendants” of that element. Parents of an element and their parents, and so
235 on, will be collectively referred to as “ancestors” of that element.

- 236 • element owning an attribute (Attribute[owner element])

237 The owner of an attribute is the element that possesses or contains the attribute.

238 The use of the term “document element” from **[XMLInfoSet]**, to describe the root of all
239 elements in an XML document, is preferred over the informal and non-standard term “root
240 element.”

241 1.4.3. XML Schema Terminology

242 The terms “W3C XML Schema”, “XML Schema” (upper case “Schema”) and “XSD” all
243 refer to the XML Schema specification, Parts 1 and 2 of the *W3C XML Schema Definition*
244 *Language (XSD) Recommendations* (**[XMLSchemaStructures]** and
245 **[XMLSchemaDatatypes]**).

246 The term “XML schema” (lower case “schema”) refers to specific XML schema
247 documents that conform to the XML Schema specifications listed above.

248 The term “XML instance” refers to an XML instance document, which is defined by and
249 validates to a particular XML schema.

250 The term “schema component” is defined in **[XMLSchemaStructures]** as a building
251 block for XML Schema. This document refers to, rather than restates, the definitions to
252 the different schema components associated with the XML Schema Abstract Data Model,
253 which are defined in the XML Schema specification. In this document, the name of the
254 referenced schema component may appear without the suffix “schema component” (i.e.
255 the term “complex type definition” is used instead of “complex type definition schema
256 component”), to enhance readability of the text.

257 The term “NCName” is defined in **[XMLSchemaDatatypes]**, and refers to XML “non-
258 colonized” names, i.e., XML name strings that do not contain the “:” character.

259 1.4.4. XML Namespace Terminology

260 This document uses the concept of an “XML Namespaces” as defined by
261 **[XMLNamespaces]** and **[XMLNamespacesErrata]**.

262 1.5. Document Organization

263 This remainder of this document is organized into sections as follows:

- 264 • The NIEM Conceptual Model discusses the underlying semantic model for NIEM.

265	• Guiding Principles discusses the principles which serve as the foundation and
266	guidelines for the rules.
267	• Relation to Standards discusses the use of the key standards used in the
268	development of NIEM.
269	• XML Schema Design Rules discusses the rules for using XML Schema
270	constructs in NIEM-conformant schemas.
271	• Modeling Rules discusses the rules for additional structure and constraints
272	needed to build NIEM-conformant schemas.
273	• XML Instance Rules discusses the rules for NIEM-conformant XML instance
274	documents.
275	• Naming Rules discusses the rules used in naming NIEM-conformant data
276	components.
277	NOTE: The ordering of the sections is intended to minimize the number of forward
278	references in the document. For this reason, the naming rules appear as the last section
279	of the document, so that the concepts being named have already been discussed.
280	This document also contains appendices of reference material as follows:
281	• A brief, non-normative overview of NIEM.
282	• A listing of all design principles, for reference purposes.
283	• A listing of all rules, for reference purposes.
284	• A table summarizing the NIEM names syntax for special NIEM components.
285	• Tables that appear in the body of this document, repeated for reference
286	purposes.
287	• Discussion and full listings of the NIEM 2.0 supporting schemas (<code>structures</code>
288	and <code>appinfo</code>).
289	• An itemized listing of the NIEM 2.0 reference schemas.
290	• A listing of high level design guidelines.
291	• A listing of modeling guidelines for harmonization.
292	• References to external standard documents.
293	A glossary of all the normative definitions found throughout this document, for
294	reference purposes.
295	

296 **2. The NIEM Conceptual Model**

297 The NIEM provides a concrete semantic model, leveraging concepts from XML Schema,
298 RDF and the ISO/IEC Standard 11179 Metadata Registries. This semantic model
299 underlies all NIEM-conformant schemas, as well as NIEM-conformant instance data.
300 XML data that follows the rules of NIEM imply specific meaning. The XML Schema
301 components used in NIEM are selected to clarify the meaning of XML data.

302 NIEM provides a framework, within which XML data may be understood to have specific
303 meaning. In general, one limitation of XML is that it does not describe the meaning of an
304 XML document. NIEM adds to the XML specification a guide to determining the meaning
305 of any given document.

306 The goal of this section is to clarify the meaning of XML data conformant to NIEM, and to
307 outline the implications of various modeling constructs in NIEM. The NIEM follows, at a
308 high level, the RDF conceptual model **[RDFConcepts]**, as outlined in this section.

309 The rules for NIEM-conformant schemas and instances are in place to ensure that a
310 specific meaning can be derived from data. That is, the data makes specific assertions,
311 and those assertions are well-understood, since they are derived from the rules for NIEM.

312 The key concepts underpinning the NIEM Conceptual Model are discussed in the
313 remainder of this section:

- 314 • NIEM Data Objects
- 315 • NIEM Data Assertions
- 316 • NIEM Data Model Explicit Not Implicit
- 317 • NIEM Data Model Implementation in XML Schema

318 **2.1. NIEM Data Objects**

319 In NIEM, an exchange is generally ad-hoc. That is, a message may be generated
320 without any persistence. It exists only for the purpose of exchange, and may not have
321 any universal meaning beyond the specific exchange. As such, a message may or may
322 not have a URI as an identifier. NIEM was designed with the assumption that a given
323 exchange may not have any unique identifier. This differs from RDF, in which all entities
324 (other than literal values) are identified by globally-meaningful URIs.

325 In NIEM, an object (data instance) is assumed to not be identified by a URI. This differs
326 from RDF, where each data object is identified by its URI. In NIEM, there is not
327 necessarily a universal, unique identifier for any given data object.

328 A NIEM-conformant instance uses XML IDs to identify objects within an XML document,
329 The NIEM XML ID is an attribute `structures:id`, of type `xsd:ID`. These IDs are not
330 assumed by NIEM to have any universal significance; they need only be unique within
331 the XML document. The use of an ID is required only when an object must be referenced
332 within the document. NIEM recognizes no correlation between these local IDs and any
333 URI.

334 Any given implementation, message, or IEPD may be defined to apply a URI or other
335 universally-meaningful identifier to an object or message. However, NIEM has no such
336 requirement.

337 Objects are instances of classes, in an object-oriented design sense. In RDF, such
338 classes are described by types, which is also how NIEM refers to them. In RDF, a
339 schema describes these classes. NIEM represents classes with type definition
340 definitions: complex type definitions and simple type definitions.

341 Data describes characteristics of objects and relationships between objects. In RDF,
342 these characteristics and relationships are called **properties** of objects, which is also
343 how NIEM refers to them. NIEM represents properties with element declarations and
344 attribute declarations.

345 Within data, an instance of a property has several characteristics. The terminology
346 comes from RDF, and is similar to the words describing the relationship of a verb to
347 nouns in a sentence: a verb has a subject and an object.

- 348 • The **property** itself: What relationship is being asserted? For example, the
349 property may say that there are brothers, or that someone has hair of a particular
350 color.
- 351 • The **subject**: About what object is the property being asserted? This would be
352 the person that has the brother, or the person whose hair is being described.
- 353 • The **object**: What is the value of the property, or to what other object does the
354 relationship exist? This would be the person that is the brother of the subject, or
355 person whose hair has the color brown.

356 A property relates *two* objects. Data will describe an object having a characteristic with a
357 specific value, or will describe an object with a particular relationship to another object.
358 All properties are pair-wise: between two objects, or between an object and a value.

359 In theory, any relationship that involves more than two objects may be modeled as a set
360 of binary properties. In NIEM, such relationships may be expressed either as a set of
361 properties (i.e. as element and attribute declarations) or as a complex type definition.

362 2.2. NIEM Data Assertions

363 Data consists of **assertions** about objects. These assertions are categorized as follows:

- 364 • Assertions that **objects exist**
365 Any reference to a data object asserts that the object exists. For example, XML
366 data about a person says that the person exists.
- 367 • Assertions that **objects have characteristics**
368 Any reference to some characteristic of the object. For example, XML data about
369 a person with the name "John" asserts that a person has a characteristic called
370 "name" and the characteristic has a value of "John."
- 371 • Assertions that **objects participate in relationships**
372 Any reference to relationship from one object to one or more objects. For
373 example, XML data about a person may contain a characteristic which represents
374 a "brother" relationship. The value of that characteristic refers to another object
375 that is considered to be a person. The XML data associated with the person
376 assert that there is a person, that the person is in a relationship with another
377 person, and that these two people are brothers.

378 2.3. NIEM Data Model Explicit Not Implicit

379 In NIEM data, that which is not stated is not implied. If data says a person's name is
380 "John", it is not implicitly saying that he doesn't have other names, or that "John" is his
381 legal name, or that he is different from a person known as "Bob." The only assertion
382 being made is that one of the names by which this person is known is "John".

383 This is one reason that definitions of NIEM content are so important. The definitions must
384 state exactly what is implied by any given statement. The concept of "legal name" may
385 be defined that makes additional assertions about a name of a person. Such assertions
386 must be made explicit in the definition of the relationship.

2.4. NIEM Data Model Implementation in XML Schema

NIEM defines rules for XML schemas which enforce the NIEM conceptual model. The schemas which follow these rules are referred to as **NIEM-conformant schemas**.

As discussed above, NIEM classes and properties are mapped onto XML Schema components. The following is an example of how a NIEM class for “Person” is rendered as an XML Schema complex type definition:

Conceptual class rendered as XML Schema complex type

```
<xsd:complexType name="PersonType">
...
</xsd:complexType>
```

The following is an example of how a NIEM property for “ImageOperator” is rendered as an element declaration:

Conceptual property rendered as element declaration

```
<xsd:element name="ImageOperator" type="nc:PersonType" nillable="true">
...
</xsd:element>
```

NIEM also defines rules for XML documents which enforce the NIEM conceptual model. XML data is called a **NIEM-conformant instance** if it follows the rules specified by the NIEM-conformant schema, as well as additional rules that are NIEM-specific. For example, in a NIEM-conformant instance, XML IDREFs must refer to XML IDs defined on objects of appropriate type. If this is not the case, the data may be valid according to the XML schema, but will not be NIEM-conformant.

Sample fragment of NIEM-conformant data

```
<nc:Person>
  <nc:PersonHairColorCode>BRN</nc:PersonHairColorCode>
</nc:Person>
```

Based on an element declaration from NIEM Core, the following example illustrates a valid XML instance that does not conform to NIEM. Per the `appinfo:ReferenceTarget` element in the schema declaration, `nc:ActivityReference` may **ONLY** refer to an `nc:ActivityType`. However, within the instance, `my:ActivityList/nc:ActivityReference` refers to “Bill”, which is an `nc:PersonType`.

420

Schema declaration for element nc:ActivityReference

421

422

423

424

425

426

427

428

429

430

```
<xsd:element name="ActivityReference" type="structures:ReferenceType">
  <xsd:annotation>
    <xsd:documentation>
      A single or set of related actions, events, or process steps.
    </xsd:documentation>
    <xsd:appinfo>
      <appinfo:ReferenceTarget appinfo:name="ActivityType"/>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>
```

431

Valid instance for above schema that does NOT conform to NIEM rules

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

```
<nc:Person structures:id="Bill">
  <nc:PersonFullName>William Tell</nc:PersonFullName>
  <nc:PersonSexCode>M</nc:PersonSexCode>
</nc:Person>

<nc:Activity structures:id="Pie">
  <nc:ActivityDescriptionText>
    County fair pie-eating contest
  </nc:ActivityDescriptionText>
</nc:Activity>

<my:ActivityList>
  <nc:ActivityReference structures:ref="Pie"/>
  <nc:ActivityReference structures:ref="Bill"/>
</my:ActivityList>
```

447 **3. Guiding Principles**

448 Principles in this specification provide a foundation for the rules. These principles are
449 generally applicable in most cases. They should not be used as a replacement for
450 common sense or appropriate special cases.

451 The principles are not operationally enforceable; they do not specify constraints on XML
452 schemas and instances. The rules are the normative and enforceable manifestation of
453 the principles.

454 The principles discussed in this section are categorized as follows:

- 455 • Specification Guidelines
- 456 • XML Schema Design Guidelines
- 457 • Modeling Design Guidelines
- 458 • Implementation Guidelines

459 **3.1. Specification Guidelines**

460 The principles in this section address what material should be included in this NDR, and
461 how it should be represented.

462 **3.1.1. Keep Specification To Minimum**

463 This specification should state what is required for interoperability, not all that could be
464 specified. Certain decisions (such as normative XML comments) could create roadblocks
465 for interoperability, making heavy demands on systems for very little gain. The goal is not
466 standardization for standardization's sake. The goal is to maximize interoperability and
467 reuse.

468 **[Principle 1]**

469 This specification should specify what is necessary for interoperability, and no
470 more.

471 **3.1.2. Focus On Rules For Schemas**

472 This specification should try, as much as is possible, to specify schema-level content.
473 This is a specification for schemas, and so should specify schemas. It should avoid
474 specifying complex data models, or data dictionaries.

475 **[Principle 2]**

476 This specification should focus on providing rules for specifying schemas.

477 **3.1.3. Use Specific Concise Rules**

478 A rule should be as precise and specific as possible, to avoid broad, hard-to-modify rules.
479 Putting multiple clauses in a rule makes it harder to enforce. Using separate rules allows
480 specific conditions to be clearly stated.

481 **[Principle 3]**

482 This specification should feature rules which are as specific, precise, and concise
483 as possible.

484 **3.2. XML Schema Design Guidelines**

485 The principles in this section address how XML Schema technology should be used in
486 designing NIEM-conformant schemas and instances.

487 **3.2.1. Disallow Content Modification with XML Processors**

488 XML Schema has constructs that can make the data provided by XML processors
489 different before and after schema processing. A sample of this is the use of XML Schema
490 attribute declarations with default values. Before XML schema validation, there may be
491 no attribute value, but after processing, the attribute value exists.

492 Within NIEM, the purpose of processing instances against schemas is solely validation:
493 testing that data instances match desired constraints and guidelines. It should not be
494 used to change the content of data instances.

495 **[Principle 4]**

496 The content of a NIEM-conformant data instance should not be modified by
497 processing against XML schemas.

498 **3.2.2. Use XML Validating Parsers for Content Validation**

499 NIEM is designed for XML Schema validation. A primary goal is to maximize the amount
500 of validation that may be performed by XML Schema validating parsers.

501 XML Schema validates content using content models: descriptions of what elements and
502 attributes may be contained within an element, and what values are allowable.

503 Mechanisms involving linking using attribute and element values are useful, but should
504 only be relied upon when absolutely necessary.

505 **[Principle 5]**

506 NIEM should depend on XML Schema validating parsers for validation of XML
507 content.

508 **3.2.3. Validate for Conformance to Reference Schemas**

509 Systems that operate on XML data have the opportunity to perform multiple layers of
510 processing. Data may be processed by middleware, XML libraries, XML schemas, and
511 application software.

512 **[Principle 6]**

513 The primary purpose of XML Schema validation is to restrict processed data to
514 that data that conforms to agreed-upon rules. This restriction is achieved by
515 marking as invalid that data that does not conform to the rules defined by the
516 schema.

517 **3.2.4. Allow Multiple Schemas for XML Constraints**

518 The NIEM does not attempt to create a one-size-fits-all schema, to perform all validation.
519 Instead, it creates a set of reference schemas, on which additional constraints may be
520 placed. It also does not focus on language-binding XML Schema implementations, which
521 convert XML Schema definitions into working programs. It is, instead, focused on
522 normalizing language and preserving the meaning of data.

523 **[Principle 7]**

524 Constraints on XML instances MAY be validated by multiple schema validation
525 passes, using multiple schemas for a single namespace.

526 **3.2.5. Define One Reference Schema Per Namespace**

527 NIEM uses the concept of a *reference schema*, which defines the structure and content
528 of a namespace. For each NIEM-conformant namespace, there is exactly one NIEM
529 reference schema. A user may use a NIEM subset schema in place of a NIEM reference

530 schema, but all NIEM-conformant instances must validate against a single reference
531 schema for each namespace.

532 **[Principle 8]**

533 Each NIEM-conformant namespace will be defined by exactly one reference
534 schema.

535 **3.2.6. Disallow Mixed Content**

536 When validating XML instance data against XML schemas, mixed content is very difficult
537 to constrain. Instances that use mixed content are difficult to specify, and complicate the
538 task of data processing. Much of the payload carried by mixed content is unchecked, and
539 does not facilitate data standardization or validation.

540 **[Principle 9]**

541 NIEM-conformant schemas do not specify data that uses mixed content.

542 **3.2.7. Specify Types for All Constructs**

543 Schema components within NIEM all have names. This means that there are no
544 anonymous types, elements, or other components defined by NIEM. Once an application
545 has determined the name (i.e. namespace and local name) of an attribute or element
546 used in NIEM-conformant instances, it will also know the type of that attribute or element.

547 There are no local attributes or elements defined by NIEM, only global attributes and
548 elements. This maximizes the ability of application developers to extend, restrict, or
549 otherwise derive definitions of local components from NIEM-conformant components.

550 **[Principle 10]**

551 Using named global components in schemas maximizes the capacity for reuse.

552 **3.2.8. Avoid Wildcards In Reference Schemas**

553 Wildcards in NIEM-conformant schemas work in opposition to standardization. The goal
554 of creating harmonized, standard schemas is to standardize definitions of data. The use
555 of wildcard mechanisms (such as `xsd:any`, which allows insertion of an arbitrary number
556 of elements from any namespace) allow non-standard data to be passed via otherwise
557 standardized exchanges. Avoidance of wildcards encourages the separation of
558 standardized and non-standardized data. It encourages users to incorporate their data
559 into NIEM in a standardized way. It also encourages users to extend in a way that may
560 be readily incorporated into NIEM.

561 **[Principle 11]**

562 Wildcards in standard schemas should be avoided.

563 **3.2.9. Provide Default Reference Schema Locations**

564 **[XMLSchemaStructures]** provides three ways to specify the physical location of an XML
565 schema: `schemaLocation`, an attribute of the element `xsd:import`, along with
566 `xsi:schemaLocation` and `xsi:noNamespaceSchemaLocation`, attributes of an
567 XML schema document element. In all of these uses, the specification explicitly
568 maintains that the schema location specified is a hint, which may be overridden by
569 applications.

570 **[Principle 12]**

571 Schema locations specified within NIEM-conformant reference schemas are hints
572 and provide default values to processing applications.

573 3.3. Modeling Design Guidelines

574 The principles in this section address the design philosophy used in designing the NIEM
575 conceptual model.

576 3.3.1. Namespaces Enhance Reuse

577 NIEM is designed to maximize reuse of namespaces and the schemas that define them.
578 When referring to a concept defined by NIEM, a user should ensure that instances and
579 schemas refer to the namespace defined by NIEM. User-defined namespaces should be
580 used for specializations and extension of NIEM constructs, but should not be used when
581 the NIEM structures are sufficient.

582 [Principle 13]

583 NIEM-conformant instances and schemas should reuse components from NIEM
584 distribution schemas when possible.

585 NIEM relies heavily on XML namespaces to prevent naming conflicts and clashes.
586 Reuse of any component is always by reference to both its namespace and its local
587 name. All NIEM component names have global scope, therefore validation always
588 occurs against the reference schemas or subsets thereof.

589 Example:

```
590 <xsd:element ref="nc:BinaryCaptureDate"  
591 minOccurs="0"  
592 maxOccurs="unbounded"/>
```

593 In this example, `nc:BinaryCaptureDate` is reused by referencing its element
594 declaration through both its namespace (which is bound to the prefix `nc:`) and its local
595 name (`BinaryCaptureDate`). If an element named `BinaryCaptureDate` is declared
596 in another namespace, it is an entirely different element and is unrelated to
597 `nc:BinaryCaptureDate`. There is no implicit relationship to
598 `nc:BinaryCaptureDate`. Any relationship must be made explicit using methods
599 outlined in this document.

600 [Principle 14]

601 A namespace is a required part of the name of a component. A component's
602 local name is considered independent of, and unassociated with, names from
603 other namespaces.

604 3.3.2. Design NIEM for Extensibility

605 NIEM is designed to be extended. Numerous methods are considered acceptable in
606 creating extended and specialized components.

607 [Principle 15]

608 NIEM is intended for extension and augmentation by users and developers
609 outside the standardization process.

610 3.4. Implementation Guidelines

611 The principles in this section address issues pertaining to the implementation of
612 applications that use NIEM.

613 **3.4.1. Avoid Displaying Raw XML Data**

614 XML data should be made human-understandable when possible, but it is not targeted at
615 human consumers. XML Schema is intended for validators and automatic processing.
616 HTML is intended for browsers. Browsers and similar technology provide human
617 interfaces to XML and other structured content. As such, structured XML content does
618 not belong in places targeted towards human consumption. Human-targeted information
619 should be of a form suitable for presentation.

620 **[Principle 16]**

621 XML data is primarily intended for automatic processing, not for literal
622 presentation to people.

623 **3.4.2. Leave Implementation Decisions To Implementers**

624 NIEM is intended to be an open specification, supported by many diverse
625 implementations. It was designed from data requirements and not from or for any
626 particular system or implementation. Use of NIEM should not depend on specific
627 software, other than XML Schema validating parsers.

628 **[Principle 17]**

629 NIEM should not depend on specific software packages, frameworks, or systems
630 for interpretation of XML instances.

631 Similarly, the NIEM should be implemented with commercial off-the-shelf and free
632 software products.

633 **[Principle 18]**

634 NIEM should be implemented with a variety of commercial off-the-shelf and free
635 software products.

636 **3.4.3. Documentation**

637 As will be described in later sections of this document, all NIEM components are
638 documented through their definitions and names. Although it is often very difficult to
639 apply, a data component definition should be drafted before the data component name is
640 assigned.

641 Drafting the definition for a data component first, ensures that the author understands the
642 exact nature of the entity or concept that the data component represents. The
643 component name should subsequently be composed to summarize the definition.
644 Reversing this sequence often results in data definitions that very precisely describe the
645 component name, but do not adequately describe the entity or concept that the
646 component is designed to represent. This potentially leads to the ambiguous use of such
647 components.

648 **[Principle 19]**

649 A data component definition should be drafted before the associated data
650 element name is composed.

651 **3.4.4. Consistent Naming**

652 Components in NIEM should be given names which are consistent with names of other
653 NIEM components. Having consistent names for components has several advantages:

- 654 1. It is easier to determine the nature of a component when it has a name
655 that conveys the meaning and use of the component.
- 656 2. It is easier to find a component when it is named predictably.

657 3. It is easier to create a name for a component when clear guidelines exist.

658 **[Principle 20]**

659 Components in NIEM should be given names which are consistent with names of
660 other NIEM components. Such names should be based on simple rules.

661 4. Relation to Standards

662 This section specifies the standards and specifications to which the NIEM conforms.
663 Where NIEM differs from public standards, the rationale for those differences is
664 discussed in this section. The complete list of standards and specifications referenced in
665 this section appears in Appendix I, References.

666 4.1. XML 1.0

667 [Rule 4-1]

668 A NIEM-conformant schema **MUST** conform to XML as specified by **[XML]**.

669 Rationale

670 XML is a well-known, commonly used W3C Recommendation. It is supported by
671 a large number of commercial and open source software tools. It is a simple,
672 well-defined, semi-structured data format that is flexible enough to allow for easy
673 extension. XML works with many other powerful associated technologies such
674 as XSLT and XPath. Artifacts of NIEM conform to the most recent
675 recommendation for XML.

676 4.2. XML Namespaces

677 [Rule 4-2]

678 A NIEM-conformant schema **MUST** conform to the specification for namespaces
679 in XML, as defined by **[XMLNamespaces]** and **[XMLNamespacesErrata]**.

680 Rationale

681 NIEM is designed to facilitate cross-domain data exchanges and interoperability.
682 The ultimate scope of NIEM is anticipated to be quite large. The primary purpose
683 of namespaces is to avoid naming conflicts, which for NIEM could become quite
684 common, since NIEM stakeholders and IEPD developers define and name many
685 of their own data components independently. Therefore, in NIEM, XML
686 namespaces are employed both to avoid name clashes and to provide a level of
687 independence to participating domains.

688 4.3. XML Schema

689 [Rule 4-3]

690 A NIEM-conformant schema **MUST** conform to the W3C XML Schema
691 Recommendations: XML Schema Part 1: Structures and XML Schema Part 2:
692 Datatypes, as specified by **[XMLSchemaStructures]** and
693 **[XMLSchemaDatatypes]**.

694 Rationale

695 XML Schema has become the generally accepted schema language, and is
696 experiencing the most widespread adoption. Although other schema languages
697 exist that offer their own advantages and disadvantages, the current approach is
698 to base NIEM on XML Schema.

699 4.4. ISO 11179, Part 4

700 Good data definitions are fundamental to data interoperability. You cannot effectively
701 exchange what you cannot understand. NIEM employs the guidance of **[ISO 11179 Part**
702 **4]** as a baseline for its data component definitions. All NIEM components are
703 documented.

704 **[Definition: documented component]**

705 In a NIEM-conformant schema, a **documented component** is an XML Schema
706 component that is required to have associated documentation. These schema
707 components are required to have a textual definition for the component to be
708 well-understood. Schemas that do not document their components accordingly
709 are not NIEM-conformant.

710 **[Definition: definition]**

711 The **definition** of a documented component is the content of the occurrence of
712 an element `xsd:documentation` that is an immediate child of the occurrence
713 of an element `xsd:annotation`. That element `xsd:annotation` is itself an
714 immediate child of the element that defines the component.

715 **Example of definition of MeasureMetadataType**

```
716 <xsd:complexType name="MeasureMetadataType">
717   <xsd:annotation>
718     <xsd:documentation>
719       A data type for metadata about a measurement.
720     </xsd:documentation>
721     <xsd:appinfo>
722       <appinfo:Base
723         appinfo:namespace=http://niem.gov/niem/structures/2.0
724         appinfo:name="MetadataType"/>
725       <appinfo:AppliesTo appinfo:name="MeasureType"/>
726     </xsd:appinfo>
727   </xsd:annotation>
728   <xsd:complexContent>
729     <xsd:extension base="s:MetadataType">
730       <xsd:sequence>
731         <xsd:element ref="nc:MeasureDate"
732           minOccurs="0" maxOccurs="unbounded"/>
733         <xsd:element ref="nc:Measurer"
734           minOccurs="0" maxOccurs="unbounded"/>
735       </xsd:sequence>
736     </xsd:extension>
737   </xsd:complexContent>
738 </xsd:complexType>
```

739 **[Rule 4-4]**

740 Within a NIEM-conformant schema, the text definition provided for each
741 documented component SHALL follow the requirements and recommendations
742 for data definitions given by **[ISO 11179 Part 4]**.

743 **Rationale**

744 To advance the goal of creating semantically-rich NIEM-conformant schemas, it
745 is necessary that data definitions be descriptive, meaningful, and precise. **[ISO**
746 **11179 Part 4]** provides standard structure and rules for defining data definitions.
747 The NIEM uses this standard for component definitions.

748 Note that the metadata maintained for each NIEM component contains additional details,
749 including domain-specific usage examples and keywords. Such metadata is used to
750 enhance search and discovery of components in a registry, and therefore, is not included
751 in schemas.

752 For convenience and reference, the summary requirements and recommendations in
753 **[ISO 11179 Part 4]** are reproduced here:

754 **ISO 11179 Requirements**

755

- 756 A data definition SHALL:
- 757 • be stated in the singular.
 - 758 • state what the concept is, not only what it is not.
 - 759 • be stated as a descriptive phrase or sentence(s).
 - 760 • contain only commonly understood abbreviations.
 - 761 • be expressed without embedding definitions of other data or underlying concepts.

762

763 **ISO 11179 Recommendations**

764

765 A data definition SHOULD:

- 766 • state the essential meaning of the concept.
- 767 • be precise and unambiguous.
- 768 • be concise.
- 769 • be able to stand alone.
- 770 • be expressed without embedding rationale, functional usage, or procedural
- 771 information.
- 772 • avoid circular reasoning.
- 773 • use the same terminology and consistent logical structure for related definitions.
- 774 • be appropriate for the type of metadata item being defined.

775 In addition to the requirements and recommendations of **[ISO 11179 Part 4]**, NIEM also
776 applies additional rules to data definitions. These rules are detailed in Section 6.2.1,
777 Human-Readable Documentation.

778 **4.5. ISO 11179, Part 5**

779 Names are a simple but incomplete means of providing semantics to data components.
780 Data definitions, structure, and context help to fill the gap left by the limitations of naming.
781 The goals for data component names should be syntactic consistency, semantic
782 precision, and simplicity. In many cases, these goals conflict and it is sometimes
783 necessary to compromise or to allow exceptions to ensure clarity and understanding. To
784 the extent possible, NIEM applies **[ISO 11179 Part 5]** to construct NIEM data component
785 names.

786 The set of NIEM data components is a collection of data representations for real world
787 objects, concepts, their associated properties and relationships. Thus, names for these
788 components would consist of the terms (words) for object classes or that describe object
789 classes, their characteristic properties, subparts, and relationships.

790 **[Rule 4-5]**

791 In general, a NIEM component name SHALL be formed by applying the
792 informative guidelines and examples detailed in Annex A of **[ISO 11179 Part 5]**,
793 with exceptions as specified in this document, most notably those specified in
794 Section 8, Naming Rules.

795 **Rationale**

796 The guidelines and examples of **[ISO 11179 Part 5]** provide a simple, consistent
797 syntax for data names which captures context and thereby imparts a reasonable
798 degree of semantic precision.

799 NIEM uses the guidelines and examples of **[ISO 11179 Part 5]** as a baseline for
800 normative naming rules. However, some NIEM components require bending of these
801 rules. Special naming rules for these classes of components are presented and
802 discussed in Section 8. In spite of these exceptions, most NIEM component names can
803 be disassembled into their **[ISO 11179 Part 5]** constituent words or terms.

804 **Example:**

805 The NIEM component name `AircraftFuselageColorCode` disassembles as follows:

- 806 • Object class term = "Aircraft"
- 807 • Qualifier term = "Fuselage"
- 808 • Property term = "Color"
- 809 • Representation term = "Code"

810 Section 8, Naming Rules details the specific rules for each kind of term and how to
811 construct NIEM component names from them. Exceptions for special components are
812 also described in Section 8.

813 5. XML Schema Design Rules

814 The W3C XML Schema Language provides many features that allow a developer to
815 represent a logical data model many different ways. This section establishes rules for the
816 use of XML Schema constructs within NIEM-conformant schemas. Because the XML
817 Schema specifications are flexible, comprehensive rules are needed to achieve a
818 balance between establishing uniform schema design and providing developers flexibility
819 to solve novel data modeling problems

820 Note that external schemas (non NIEM-conformant schemas) do not need to obey the
821 rules set forth in this section. So long as schema components from external schemas are
822 adapted for use with NIEM, according to the modeling rules in Section, , they may be
823 used as they appear in the external standard, even if the schema components violate the
824 rules for NIEM-conformant schemas.

825 The XML Schema design rules in this section fall into the following categories:

- 826 • Restrictions on XML Schema Constructs
- 827 • `xsd:schema` Document Element
- 828 • Namespace Imports
- 829 • Annotations
- 830 • Type Definitions
- 831 • Additional De

832 5.1. Restrictions on XML Schema Constructs

833 There are a number of XML Schema constructs that are not used within NIEM-
834 conformant schemas. Many of these constructs provide capability that is not currently
835 needed within NIEM. Some of these constructs create problems for interoperability, or
836 with tool support, or with clarity or precision of data model definition.

837 5.1.1. No Mixed Content

838 [Rule 5-1]

839 Within a NIEM-conformant schema, an element `xsd:complexType` SHALL
840 NOT own the attribute `mixed` with the value `true`.

841 [Rule 5-2]

842 Within a NIEM-conformant schema, an element declaration which is of complex
843 content SHALL NOT own the attribute `mixed` with the value `true`.

844 Rationale

845 Mixed content allows the mixing of data tags with text. Languages such as
846 XHTML use this syntax for markup of text. NIEM-conformant schemas define
847 XML that is for data exchange, not text markup. Mixed content creates
848 complexity in processing, defining, and constraining content.

849 Well-defined markup languages exist outside of NIEM, and may be used with
850 NIEM data. External schemas may include mixed content, and may be used with
851 NIEM. However, mixed content must not be defined by NIEM-conformant
852 schemas in keeping with [Principle 9].

853 **5.1.2. No Notations**

854 **[Rule 5-3]**

855 A NIEM-conformant schema SHALL NOT contain a reference to the type
856 definition `xsd:NOTATION`, or to a type derived from that type.

857 **[Rule 5-4]**

858 A NIEM-conformant schema SHALL NOT contain the element `xsd:notation`.

859 **Rationale**

860 XML Schema notations allow the attachment of system and public identifiers on
861 fields of data. The notation mechanism does not play a part in validation of
862 instances and is not supported by NIEM.

863 **5.1.3. No Schema Inclusion**

864 **[Rule 5-5]**

865 A NIEM-conformant schema SHALL NOT contain the element `xsd:include`.

866 **Rationale**

867 Element `xsd:include` brings schemas defined in separate files into the current
868 namespace. It breaks a namespace up into arbitrary partial schemas, which
869 needlessly complicates the schema structure, making it harder to reuse, and
870 process, and also increases the likelihood of conflicting definitions.

871 Inclusion of schemas that don't have namespaces also complicates schema
872 understanding. This inclusion makes it difficult to find the realization of a specific
873 schema artifact, and creating aliases for schema components that should be
874 reused. Inclusion of schemas also violates **[Principle 8]**, as it uses multiple
875 schemas to construct a namespace.

876 **5.1.4. No Schema Redefinition**

877 **[Rule 5-6]**

878 A NIEM-conformant schema SHALL NOT contain the element `xsd:redefine`.

879 **Rationale**

880 The `xsd:redefine` element allows an XML schema to restrict and extend
881 components from a namespace, in that very namespace. Such redefinition
882 introduces duplication of definitions, allowing multiple definitions to exist for
883 components from a single namespace. This violates **[Principle 8]** that a single
884 reference schema defines a NIEM-conformant namespace.

885 **5.1.5. Wildcard Restrictions**

886 There are many constructs within XML Schema that act as wildcards. That is, they
887 introduce buckets which may carry arbitrary or otherwise non-validated content. Such
888 constructs violate **[Principle 11]**, and as such provide implicit workarounds for the difficult
889 task of agreeing on the content of data models. Such workarounds should be made
890 explicitly, outside the core data model.

891 **5.1.5.1. No Unconstrained Type Substitution**

892 **[Rule 5-7]**

893 A NIEM-conformant schema SHALL NOT reference the type `xsd:anyType`.

894 **Rationale**

895 XML Schema has the concept of the "ur-type", a type that is the root of all other
896 types. This type is realized in schemas as `xsd:anyType`.

897 NIEM-conformant schemas must not use `xsd:anyType`, because this feature
898 permits the introduction of arbitrary content (i.e. untyped and unconstrained data)
899 into an XML instance. NIEM intends that all constructs within the instance be
900 described by the schemas describing that instance.

901 **5.1.5.2. No Unconstrained Text Substitution**

902 **[Rule 5-8]**

903 A NIEM-conformant schema SHALL NOT reference the type
904 `xsd:anySimpleType`.

905 **Rationale**

906 XML Schema provides a restriction of the "ur-type", which contains only simple
907 content. This provides a wildcard for arbitrary text. It is realized in XML Schema
908 as `xsd:anySimpleType`.

909 NIEM-conformant schemas must not use `xsd:anySimpleType` because this
910 feature is insufficiently constrained to provide a meaningful starting point for
911 content definitions. Instead, content should be based on one of the more
912 specifically-defined simple types defined by XML Schema.

913 **5.1.5.3. Untyped Elements Must be Abstract**

914 **[Rule 5-9]**

915 Within a NIEM-conformant schema, an element declaration with the attribute
916 `name` and without the attribute `type` MUST carry the attribute `abstract` with the
917 value `true`.

918 **Rationale**

919 Untyped element declarations act as wildcards that may carry arbitrary data. By
920 declaring such types abstract, NIEM allows the creation of type independent
921 semantics without allowing arbitrary content to appear in XML instances.

922 **5.1.5.4. No Untyped Attributes**

923 **[Rule 5-10]**

924 Within a NIEM-conformant schema, an attribute declaration with attribute `name`
925 MUST carry the attribute `type`.

926 **Rationale**

927 Untyped XML schema attributes allow arbitrary content, with no semantics.
928 Attributes must have a type, so that specific syntax and semantics will be
929 provided.

930 **5.1.5.5. No Unconstrained Element Substitution**

931 **[Rule 5-11]**

932 A NIEM-conformant schema SHALL NOT contain the element `xsd:any`.

933 **Rationale**

934 The `xsd:any` particle (see Model Group Restrictions for an informative definition
935 of particle) provides a wildcard which may carry arbitrary content. The particle

936 `xsd:any` may appear within constraint schemas or within other schemas that are
937 not NIEM-conformant, but is prohibited in NIEM-conformant schemas.

938 **5.1.5.6. No Unconstrained Attribute Substitution**

939 **[Rule 5-12]**

940 A NIEM-conformant schema SHALL NOT contain the element
941 `xsd:anyAttribute`.

942 **Rationale**

943 The `xsd:anyAttribute` element provides a wildcard, where arbitrary attributes
944 may appear. The element `xsd:anyAttribute` may appear within constraint
945 schemas or within other schemas that are not NIEM-conformant, but is prohibited
946 in NIEM-conformant schemas.

947 **5.1.6. Component Naming Restrictions**

948 All NIEM components must be named. That is, type definitions, and element and
949 attribute declarations must be given explicit names -- local and anonymous component
950 definition is not allowed. Note that XML Schema enforces the placement of attribute
951 group and model group definitions as top-level components, which forces the
952 components to be named.

953 **5.1.6.1. No Anonymous Type Definitions**

954 **[Rule 5-13]**

955 Within a NIEM-conformant schema, any type definition MUST appear as an
956 immediate child of the document element `xsd:schema`.

957 **Rationale**

958 NIEM does not support anonymous types in NIEM-conformant schemas. All XML
959 Schema "top-level" types (children of the document element) are required by
960 XML Schema to be named. By requiring NIEM type definitions to be top level,
961 they are forced to be named and are therefore globally reusable.

962 **5.1.6.2. No Local Element Declarations**

963 **[Rule 5-14]**

964 Within a NIEM-conformant schema, any element declaration carrying the
965 attribute `name` MUST appear as an immediate child of the document element
966 `xsd:schema`.

967 **Rationale**

968 All schema components defined by NIEM schemas must be named, accessible
969 from outside the defining schema, and reusable across schemas. Local element
970 definitions provide named elements that are not reusable outside the context in
971 which they are defined. Requiring named NIEM elements to be top level ensures
972 that they are globally reusable.

973 **5.1.6.3. No Local Attribute Definitions**

974 **[Rule 5-15]**

975 Within a NIEM-conformant schema, any attribute declaration owning the attribute
976 `name` MUST appear as an immediate child of the document element
977 `xsd:schema`.

978 **Rationale**

979 All schema components defined by NIEM schemas are named, accessible from
980 outside the defining schema, and reusable across schemas. Local attribute
981 definitions provide named attributes that are not reusable outside the context in
982 which they are defined. Requiring named NIEM attributes to be top level ensures
983 that they are globally reusable.

984 **5.1.7. No Uniqueness Constraints**

985 **[Rule 5-16]**

986 A NIEM-conformant schema SHALL NOT contain any of the elements
987 `xsd:unique`, `xsd:key`, `xsd:keyref`, `xsd:selector`, or `xsd:field`.

988 **Rationale**

989 XML Schema provides NIEM the ability to apply uniqueness constraints to
990 schema-validated content. Such mechanisms have value, but they have not
991 been included as required for NIEM. However, these elements may be used in
992 subset or constraint schemas.

993 **5.1.8. Model Group Restrictions**

994 Complex content definitions in XML Schema use model group schema components.
995 These schema components, `xsd:all`, `xsd:choice` and `xsd:sequence`, also
996 called compositors, provide for ordering and selection of particles within a model group.

997 XML Schema defines a **particle** as an occurrence of `xsd:element`, `xsd:sequence`,
998 `xsd:choice`, `xsd:any` (wildcard) and `xsd:group` (model group) within a content
999 model. For example, an `xsd:sequence` within a XML Schema complex type is a
1000 particle. An `xsd:element` occurring within an `xsd:sequence` is also a particle.

1001 **5.1.8.1. Restrictions on Particle Ordering**

1002 **[Rule 5-17]**

1003 A NIEM-conformant schema SHALL NOT contain the element `xsd:all` or the
1004 element `xsd:choice`.

1005 **Rationale**

1006 The element `xsd:all` provides a set of particles (e.g. elements) which may be
1007 included in an instance, in no particular order. The element `xsd:choice`
1008 provides an exclusive set of particles, one of which may appear in an instance.
1009 Each of these can greatly complicate processing and may provide complex
1010 regular expressions which are difficult to comprehend and satisfy. The only
1011 particle ordering mechanism allowed for use within NIEM-conformant schemas is
1012 `xsd:sequence`

1013 **5.1.8.2. No Recursively Defined Model Groups**

1014 **[Rule 5-18]**

1015 Within a NIEM-conformant schema, any immediate child of a model group
1016 `xsd:sequence` element MUST be one of `xsd:annotation`, or
1017 `xsd:element`.

1018 **Rationale**

1019 XML Schema provides the capability for model groups to be recursively defined.
1020 This means that a sequence may contain a sequence. This rule is designed to
1021 keep content models simple, comprehensive and reusable: The content of an

1022 element should boil down to a sequence of elements, defined in as
1023 straightforward a manner as is possible.

1024

1025 **5.1.8.3. Restrictions on Named Groups**

1026 **[Rule 5-19]**

1027 A NIEM-conformant schema SHALL NOT contain the element `xsd:group`.

1028 **Rationale**

1029 NIEM does not allow groups of elements to be named other than as named
1030 complex types.

1031 **5.1.8.4. Particle Cardinality Restrictions**

1032 **[Rule 5-20]**

1033 Within a NIEM-conformant schema, if the element `xsd:sequence` carries the
1034 attribute `minOccurs`, it MUST set the value for the attribute to 1.

1035 **[Rule 5-21]**

1036 Within a NIEM-conformant schema, if the element `xsd:sequence` carries the
1037 attribute `maxOccurs`, it MUST set the value of the attribute to 1.

1038 **Rationale**

1039 XML Schema allows each particle to specify cardinality (how many times the
1040 particle may appear in an instance). NIEM restricts the cardinality of
1041 `xsd:sequence` and `xsd:group` particles to exactly one, to ensure that content
1042 model definitions are defined in as straightforward a manner as possible.

1043 **Discussion**

1044 Note that the particle `xsd:any` is not allowed in NIEM-conformant schema by
1045 **[Rule 5-11]**

1046 Note also that element declarations acting as a particle (particles formed by
1047 `xsd:element`) may have any cardinality; they are not restricted by this rule.
1048 Should a user desire the behavior that would be obtained from the use of special
1049 cardinalities on these particles, he should define them within explicitly-named
1050 elements.

1051 **5.1.9. Block Substitution Restrictions**

1052 XML Schema provides a mechanism that will prevent substitution for an element
1053 declaration or type definition. That is, an element declaration may declare one or more of
1054 the following:

- 1055 1. An instance of this element declaration may not substitute an extended type
- 1056 2. An instance of this element declaration may not substitute a restricted type
- 1057 3. An instance of this element declaration may not substitute another element

1058 These restriction mechanisms are very useful in instances; they allow restriction of
1059 content models down to exact types and elements. However, in shared data models,
1060 they limit reuse and customization options, in opposition to **[Principle 13]**.

1061 **[Rule 5-22]**

1062 Within a NIEM-conformant schema, if an element declaration carries the attribute
1063 `block`, it MUST set the value for the attribute to the empty string.

1064 **[Rule 5-23]**

1065 Within a NIEM-conformant schema, if a complex type definition carries the
1066 attribute `block`, it MUST set the value for the attribute to the empty string.

1067 **[Rule 5-24]**

1068 Within a NIEM-conformant schema, if the document element `xsd:schema`
1069 carries the attribute `blockDefault`, it MUST set the value for the attribute to the
1070 empty string.

1071 **Rationale**

1072 Restriction of substitution options reduces capacity for reuse, and so is forbidden
1073 within NIEM-conformant schemas. In particular, setting the `block` value at the
1074 schema level complicates understanding of component definitions.

1075 **5.1.10. Final Value Restrictions**

1076 XML Schema provides the capability for type definitions and elements to declare a **final**
1077 value. This value prevents the creation of derived components. In shared data models,
1078 this capability limits reuse and customization options. in opposition to **[Principle 13]**

1079 **[Rule 5-25]**

1080 Within a NIEM-conformant schema, if a simple type definition carries the attribute
1081 `final`, it MUST set the value for the attribute to the empty string.

1082 **[Rule 5-26]**

1083 Within a NIEM-conformant schema, if a complex type definition carries the
1084 attribute `final`, it MUST set the value for the attribute to the empty string.

1085 **[Rule 5-27]**

1086 Within a NIEM-conformant schema, if an element declaration carries the attribute
1087 `final`, it MUST set the value for the attribute to the empty string.

1088 **[Rule 5-28]**

1089 Within a NIEM-conformant schema, if the document element `xsd:schema`
1090 carries the attribute `finalDefault`, it MUST set the value for that attribute to
1091 the empty string.

1092 **Rationale**

1093 Restriction of derivation options reduces capacity for reuse and so is forbidden
1094 within NIEM-conformant schemas.

1095 **5.1.11. Default Value Restrictions**

1096 XML Schema provides the capability for element and attribute declarations to provide
1097 default values when XML instances using those components do not provide values.

1098 **[Rule 5-29]**

1099 Within a NIEM-conformant schema, any element `xsd:element` SHALL NOT
1100 carry the attribute `default`.

1101 **[Rule 5-30]**

1102 Within a NIEM-conformant schema, any element `xsd:attribute` SHALL NOT
1103 carry the attribute `default`.

1104 **Rationale**

1105 The use of default values means that the act of validating a schema will insert a
1106 value into an XML instance where none existed prior to schema validation.
1107 Schema validation is for rejection of invalid instances, not for modifying instance
1108 content, as specified in **[Principle 4]**.

1109 **5.1.12. Simple Type Derivation Restrictions**

1110 XML Schema provides two methods for combining simple types together into more
1111 complicated simple types: NIEM explicitly disallows the use of both these methods.

1112 **5.1.12.1. No Lists of Simple Type**

1113 An `xsd:list` creates a new simple type that consists of multiple occurrences of the
1114 original type, separated by whitespaces. An example of a list of `xsd:integer` is "317
1115 4 36 114."

1116 **[Rule 5-31]**

1117 A NIEM-conformant schema SHALL NOT contain the element `xsd:list`.

1118 **Rationale**

1119 Such structured sequences of simple values should be represented with
1120 sequences of elements, rather than embedding the values in a single value.

1121 **5.1.12.2. No Unions of Simple Type**

1122 An `xsd:union` of several simple types creates a new simple type that may consist of the
1123 content of any of the member types. An example of a union is a union between
1124 `xsd:integer` and `xsd:anyURI` would produce a simple type that may contain a URI or
1125 integer value.

1126 **[Rule 5-32]**

1127 A NIEM-conformant schema SHALL NOT contain the element `xsd:union`.

1128 **Rationale**

1129 `xsd:union` loses the original semantic information associated with the member
1130 types. Providing such options should be done at the element level, rather than
1131 within the definitions of simple type.

1132 **5.2. `xsd:schema` Document Element**

1133 The features of XML Schema allow for flexibility of use for many different and varied
1134 types of implementation. NIEM requires consistent use of these features.

1135 **[Rule 5-33]**

1136 Within a NIEM-conformant schema, the document element `xsd:schema` MUST
1137 carry the attribute `targetNamespace`.

1138 **[Rule 5-34]**

1139 The value of the required attribute `targetNamespace` on the document element
1140 `xsd:schema` MUST match the production `<absolute-URI>` as defined by
1141 **[RFC3986]**.

1142 **Rationale**

1143 Schemas without defined namespaces provide definitions that are ambiguous, in
1144 that they are not universally identifiable.

1145 Absolute URIs are the only universally meaningful URIs. Finding the target
1146 namespace using standard XML Base technology is complicated, and not
1147 specified by XML Schema. Relative URIs are not universally identifiable, as they
1148 are context-specific.

1149 Discussion

1150 The document element `xsd:schema` may contain optional attributes
1151 `attributeFormDefault` and `elementFormDefault`. The values of these
1152 attributes are immaterial to a NIEM-conformant schema, as each attribute
1153 defined by a NIEM-conformant schema must be defined at the top-level, and so
1154 must be qualified with the target namespace of its declaration.

1155 [Rule 5-35]

1156 Within a NIEM-conformant schema, the document element `xsd:schema` MUST
1157 carry the attribute `version`.

1158 [Rule 5-36]

1159 The value of the required attribute `version` on the document element
1160 `xsd:schema` MUST NOT be an empty string.

1161 Rationale

1162 It is very useful to be able to tell one version of a schema from another. Apart
1163 from the use of namespaces for versioning, it is sometimes necessary to release
1164 multiple versions of schema documents. Such use might include:

- 1165 • Subset schemas
- 1166 • Error corrections or bug-fixes
- 1167 • Documentation changes
- 1168 • Contact information updates

1169 In such cases, a different value for the `version` attribute implies a different
1170 version of the schema. No specific meaning is assigned to specific version
1171 identifiers.

1172 5.3. Namespace Imports

1173 XML Schema requires that namespaces used in external references be imported using
1174 the `xsd:import` element. The `xsd:import` element appears as an immediate child of
1175 the `xsd:schema` element. A schema must import any namespace which

- 1176 1. is not the local namespace, and
- 1177 2. is referenced from the schema.

1178 The behavior of import statements is not necessarily intuitive. In short, the import
1179 introduces namespace into the schema in which the import appears; it has no transitive
1180 effect. If the namespaces of an import statement is not referenced from the schema, then
1181 the import statement has no effect. The import statement cannot be used to direct
1182 schema locations for schemas not referenced from the schema performing the import.
1183 The schema location directed by the import element may be overridden by user directive
1184 at the parser, or by being overridden by import elements from other schemas.

1185 Imports of namespaces should be made as uniform as possible; all schemas in a schema
1186 set should agree on what schema location goes with a particular namespace. Otherwise,
1187 behavior may be dependent on the behavior of the parser, and the order of components
1188 in instance documents.

1189 5.3.1. `xsd:import` Element Restrictions

1190 [Rule 5-37]

1191 Within a NIEM-conformant schema, the element `xsd:import` MUST carry the
1192 attribute `namespace`.

1193 [Rule 5-38]

1194 The value of the required attribute `namespace` carried by the element
1195 `xsd:import` MUST match the production `<absolute-URI>` as defined by
1196 [RFC3986].

1197 Rationale

1198 An import that does not specify a namespace is enabling reference to non-
1199 namespaced components. NIEM requires that all components have a defined
1200 namespace. It is important that the namespace declared by a schema be
1201 universally defined and unambiguous. Use of the standard XML Base for
1202 processing is not specified by XML Schema, and so is not supported here.

1203 [Rule 5-39]

1204 Within a NIEM-conformant schema, the element `xsd:import` MUST carry the
1205 attribute `schemaLocation`.

1206 Rationale

1207 An import that does not specify a schema location gives no clue to processing
1208 applications as to where to find an implementation of the namespace. Even
1209 though such a provided schema location may be overridden, it is important that
1210 an initial default be provided for processing.

1211 [Rule 5-41]

1212 Within a NIEM-conformant schema, the value of the required attribute
1213 `schemaLocation` carried by the element `xsd:import` MUST match either the
1214 production `<absolute-URI>`, or the definition of "*relative-path reference*", as
1215 defined by [RFC3986].

1216 Rationale

1217 The default value may be specified either as absolute or relative URIs. Since
1218 URNs are not resolvable, they are inappropriate for use in `schemaLocation`.
1219 The requirement for conformance to "*relative-path reference*" is required to avoid
1220 the more obscure syntax of "*network-path reference*" and the system-specific
1221 "*absolute-path reference*".

1222 [Rule 5-42]

1223 Within a NIEM-conformant schema, the value of the required attribute
1224 `schemaLocation` carried by the element `xsd:import` MUST be resolvable to
1225 a XML schema document file that is valid according to [XMLSchemaStructures]
1226 and [XMLSchemaDatatypes].

1227 Rationale

1228 The XML Schema specification requires that the object imported via
1229 `xsd:import` must be a schema document. This rule reinforces that
1230 requirement.

1231 **Discussion**

1232 Note that relative URI references are dereferenced from the location of the
1233 schema document performing the import, not from the location of an instance or
1234 other schema. Although NIEM distribution schemas use only relative URI
1235 references, that need not be the case for other NIEM-conformant schemas.

1236 **5.3.2. Including XML Content from Other Namespaces**

1237 Within an XML schema, there are several mechanisms to include XML content that is not
1238 from the XML or XML Schema namespaces. Those mechanisms are:

- 1239 1. Carrying attributes from other than the XML or XML Schema namespaces on an
1240 element in the XML Schema namespace.

1241 By the rules of XML Schema, any element may have attributes that are from
1242 other namespaces. These attributes do not participate in validation, but may
1243 carry information useful to tools which process schemas.

- 1244 2. Adding content to the elements `xsd:appinfo` and `xsd:documentation`.

1245 XML Schema allows arbitrary XML content to be included within annotations.
1246 Such XML does not participate in validation, but may communicate useful
1247 information to schema readers or processors.

1248 NIEM requires all such XML content to be “schema-valid.” That is, it must have a
1249 schema, and it must validate against that schema. The schemas must be introduced via
1250 `xsd:import` elements within the schema in which the content is used. This is for two
1251 reasons:

- 1252 1. Some tools require imports of namespaces used within schemas, and validate
1253 against those schemas.
1254 2. The definition and the validity of content within schemas should be clear.

1255 **[Rule 5-43]**

1256 Within a NIEM-conformant schema, when a namespace other than the XML
1257 namespace or the XML Schema namespace is used, it MUST be imported into
1258 the schema using the `xsd:import` element.

1259 **Rationale**

1260 This rule ensures that used namespaces have recognizable defining sources,
1261 and that they will cooperate with existing tools.

1262 **[Rule 5-44]**

1263 Within a NIEM-conformant schema, when a namespace other than the XML
1264 namespace or the XML Schema namespace is used, its content MUST be valid
1265 with respect to the schema imported for that namespace.

1266 **Rationale**

1267 XML Schema does not address the schema-validity of content used for
1268 annotations or attributes on schema components. This rule ensures that content
1269 used in such a manner is schema-valid. This encourages interoperable data
1270 definitions and schema documents.

1271 5.4. Annotations

1272 Annotations in XML Schema "provide for human- and machine-targeted annotations of
1273 schema components."¹ The two types: human-targeted and machine-targeted, are kept
1274 separate by the use of two separate container elements defined by XML Schema:
1275 `xsd:documentation` and `xsd:appinfo`.

1276 [Rule 5-45]

1277 Within a NIEM-conformant schema, an element SHALL have at most one
1278 instance of an element `xsd:annotation` as an immediate child.

1279 Rationale

1280 XML Schema allows annotations to be added to components in a fairly loose
1281 manner: there may be multiple annotations, each of which may have multiple
1282 `documentation` or `appinfo` elements. This flexibility in the syntax provides no
1283 additional expressivity, but does complicate processing, and so is forbidden in
1284 NIEM.

1285 5.4.1. Human-Readable Documentation

1286 XML Schema describes the content of `xsd:documentation` elements as "user
1287 information". This information is targeted for reading by humans. The XML Schema
1288 specification does not say what form human-targeted information should take. Within
1289 NIEM, user information is plain text, with no formatting or XML structure.

1290 [Rule 5-46]

1291 Within a NIEM-conformant schema, the content of an `xsd:documentation`
1292 element MUST be character information items as specified by [XMLInfoSet].

1293 Rationale

1294 According to the XML Schema specification, the content of
1295 `xsd:documentation` elements is intended for human consumption, whereas
1296 other structured XML content is intended for machine consumption. Therefore,
1297 the `xsd:documentation` element MUST NOT contain structured XML data. As
1298 such, any XML content appearing within a documentation element is in the
1299 context of human-targeted examples, and should be escaped using `<` and
1300 `>`. This rule also prohibits comments within documentation elements.

1301 See [SchemaForXMLSchema], the schema for XML Schema, as an example of
1302 documentation elements containing properly escaped XML elements.

1303 [Rule 5-47]

1304 Within a NIEM-conformant schema, the element `xsd:annotation` MUST have
1305 at most one instance of the element `xsd:documentation` as an immediate
1306 child.

1307 Rationale

1308 NIEM-conformant schemas apply specific meaning to `xsd:documentation`
1309 elements: they provide definitions for components. In this context, multiple
1310 documentation elements obscure understanding.

¹ From <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/#element-annotation>

1311 XML comments are not schema constructs and are not specifically associated
1312 with any schema-based components. As such, comments are not considered
1313 semantically meaningful by NIEM, and may not be retained through processing
1314 of NIEM schemas.

1315 **[Rule 5-48]**

1316 XML comments SHALL not be used for persistent information about constructs
1317 within XML Schemas.

1318 **Rationale**

1319 Since XML comments are not associated with any specific XML Schema
1320 construct, there is no standard way to interpret comments. As such, comments
1321 should be reserved for internal use, and XML Schema annotations should be
1322 preferred for meaningful information about components. NIEM specifically
1323 defines how information should be encapsulated in NIEM-conformant schemas
1324 via `xsd:annotation` elements.

1325 **5.4.2. Machine-Readable Annotations**

1326 XML Schema provides special annotations for support of automatic processing. The XML
1327 Schema specification provides the element `xsd:appinfo` to carry such content, and
1328 does not specify what style of content they should carry. In NIEM, `xsd:appinfo`
1329 elements carry structured XML content.

1330 **[Rule 5-49]**

1331 Within a NIEM-conformant schema, any immediate child of an `xsd:appinfo`
1332 element SHALL be an element information item, or a comment information item.

1333 **Rationale**

1334 Application information elements are intended for "automatic processing", and so
1335 should contain machine-oriented data, XML.

1336 **[Rule 5-50]**

1337 Within a NIEM-conformant schema, any element that is an immediate child of an
1338 `xsd:appinfo` element SHALL be in a namespace.

1339 **Rationale**

1340 Use of default namespace is allowed, but content has to have a real namespace,
1341 and namespaces must be declared. The XML namespaces specification
1342 includes the concept of content not in a namespace. Non-namespaced data runs
1343 counter to the principle of distinctly identifiable data definitions.

1344 **[Rule 5-50.1]**

1345 Within a NIEM-conformant schema, an element in the XML Schema namespace
1346 MUST NOT occur as a descendant of any element `xsd:appinfo`.

1347 **Rationale**

1348 NIEM-conformant schemas are designed to be very easily processed. Although
1349 uses of XML Schema elements as content of `xsd:appinfo` elements could be
1350 contrived, it is not current practice, and could seriously complicate the authoring
1351 of schema validators and processors, such as XSLT, which may evaluate XML
1352 elements by their namespace and name. Forbidding the use of XML Schema
1353 elements outside valid uses of schema will simplify such processing.

1354 5.5. Type Definitions

1355 XML Schema provides a variety of ways to define new types. This section covers first the
1356 NIEM restrictions on defining simple types and then on defining complex types, with both
1357 simple and complex content.

1358 5.5.1. Simple Type Definitions

1359 According to XML Schema, there are many ways to construct simple types. Within NIEM,
1360 the options are narrowed, in order to direct designs into fewer, better-defined patterns.

1361 [Rule 5-51]

1362 Within NIEM-conformant schemas, the element `xsd:simpleType` MUST have
1363 the element `xsd:restriction` as an immediate child.

1364 Rationale

1365 Any simple type must be a restriction of another type. The rules in Section
1366 5.1.12, Simple Type Derivation Restrictions, eliminate the use of `xsd:list` and
1367 `xsd:union` in simple type derivations. Therefore, only `xsd:restriction` may
1368 be used to make new simple types.

1369 5.5.2. Complex Type Definitions

1370 XML Schema provides a large amount of flexibility in the creation of complex types.
1371 NIEM narrows down the schema capability to a smaller set of constructs.

1372 Note that rules on prohibited constructs (Section 5.1.6.1: No Anonymous Type
1373 Definitions, above) forbid defining complex types as local types. All complex type
1374 definitions must be top-level, named components.

1375 XML Schema makes a distinction between complex types with simple content versus
1376 complex types with complex content. Complex types with simple content (CSCs) have
1377 content which is not allowed to contain XML elements. Complex types with complex
1378 content (CCCs) have content which does contain XML elements. Since mixed content is
1379 prohibited in NIEM by [Rule 5-1], all NIEM-conformant complex types are either CSCs or
1380 CCCs.

1381 [Rule 5-52]

1382 Within a NIEM-conformant schema, the element `xsd:complexType` MUST
1383 have as an immediate child either the element `xsd:complexContent` or the
1384 element `xsd:simpleContent`.

1385 Rationale

1386 XML Schema provides shorthand to defining complex content of a complex type,
1387 which is to define the complex type with immediate children which specify
1388 elements, or other groups, and attributes. In the desire to normalize schema
1389 representation of types, and to be explicit, NIEM forbids the use of that
1390 shorthand.

1391 5.5.3. Simple Content (CSC) Restrictions

1392 Within a NIEM-conformant schema, a CSC can be created one of two ways:

- 1393 1. By extension of an existing CSC, or
- 1394 2. By extension of an existing simple type.

1395 Both of these methods use the element `xsd:extension`.

1396 [Rule 5-53]

1397 Within a NIEM-conformant schema, the element `xsd:simpleContent` MUST
1398 have as an immediate child the element `xsd:extension`.

1399 **Rationale**

1400 This rule ensures that the definition of a CSC will use the XML Schema extension
1401 facility. This allows for the above cases, while disallowing much more
1402 complicated syntactic options available in XML Schema.

1403 Although the two above methods have similar syntax, there are subtle differences.
1404 NIEM's conformance rules ensure that any complex type has the necessary attributes for
1405 representing IDs, metadata, and link metadata. So, case 1 does not require adding these
1406 attributes, as they are guaranteed to occur in the base type.

1407 However, in case 2, in which a new complex type is created from a simple type, the
1408 attributes for complex types must be added. This is done by reference to the attribute
1409 group `structures:SimpleObjectAttributeGroup`:

1410 **[Rule 5-54]**

1411 Within a NIEM-conformant schema, given an element `xsd:simpleContent`
1412 with a child `xsd:extension` owning an attribute `base`, if the attribute `base` has
1413 a value that resolves to the name of a simple type, then the element
1414 `xsd:extension` MUST have an immediate child element
1415 `xsd:attributeGroup`.

1416 **[Rationale]**

1417 This rule ensures that a CSC that is created as an immediate extension of a
1418 simple type adds the attributes required for specific NIEM linking mechanisms.
1419 This creates a pattern for CSC definition as follows:

1420 **Example of CSC derived from a simple type**

```
1421 <xsd:complexType name="PercentageType">  
1422   ...  
1423   <xsd:simpleContent>  
1424     <xsd:extension base="nc:PercentageSimpleType">  
1425       <xsd:attributeGroup ref="structures:SimpleObjectAttributeGroup"/>  
1426     </xsd:extension>  
1427   </xsd:simpleContent>  
1428 </xsd:complexType>
```

1429 **5.5.4. Complex Content (CCC) Restrictions**

1430 Within a NIEM-conformant schema, a CCC can be created one of two ways:

- 1431 1. By extension of an existing complex type (CCC or CSC), or
- 1432 2. By extension of the type `structure:ComplexObjectType`

1433 Both of these methods use the element `xsd:extension`.

1434 **[Rule 5-55]**

1435 Within a NIEM-conformant schema, the element `xsd:complexContent` MUST
1436 have as an immediate child the element `xsd:extension`.

1437 **Rationale**

1438 NIEM does not support, as conformant, the use of complex type restriction.
1439 NIEM defines a language, in which specific content is allowed. It does not
1440 specify messages which forbid content. Such restrictions may be performed in

1441 non-conformant schemas, or within constraint schemas or other artifacts of
1442 constraint.

1443 Note that use of the attribute `base` on `xsd:extension` is required by XML
1444 Schema.

1445 The `xsd:extension` element says that the type under definition is an extension of
1446 another type. That type must be limited to those used with NIEM.

1447 **[Rule 5-56]**

1448 Within a NIEM-conformant schema, given an element `xsd:complexContent`
1449 with a child `xsd:extension` owning an attribute `base`, the attribute `base`
1450 MUST have a value that resolves to the name of one of

- 1451 1. the type `structures:ComplexObjectType`, or
- 1452 2. the type `structures:MetadataType`, or
- 1453 3. the type `structures:AugmentationType`, or
- 1454 4. a NIEM-conformant complex type.

1455 **[Rationale]**

1456 This rule ensures that a CCC has well-defined ancestry. In turn, this ensures that
1457 every CCC has well-defined semantics.

1458 5.6. Additional Definitions And Declarations

1459 XML Schema provides a variety of ways to declare and define elements and attributes.

1460 5.6.1. Element Declarations

1461 Within NIEM-conformant schemas, elements may be declared as abstract. Element
1462 declarations must be at the top-level, as rules in other sections prohibit the use of local
1463 elements. Elements may be defined without a type, but any element declaration that has
1464 no type must be declared abstract by **[Rule 5-9]**, which forbids anonymous type
1465 definitions.

1466 Within an element declaration, the attributes `fixed`, `nillable`, and
1467 `substitutionGroup` may be used as per the XML Schema specification. The attribute
1468 `form` is irrelevant to NIEM, as NIEM-conformant schemas may not contain local element
1469 definitions by **[Rule 5-14]**.

1470 Element uses (element declarations acting as particles) must reference top-level named
1471 elements. In an element use, NIEM allows any values for the XML Schema properties
1472 “max occurs” and “min occurs”.

1473 Based on a variety of user requirements, all elements in the NIEM 2.0 schemas are
1474 defined to allow a nil value. For example, the following XML instances are permitted in
1475 NIEM-conformant instances:

1476 `<nc:ActivityDate></nc:ActivityDate>`

1477 OR

1478 `<nc:ActivityDate/>`

1479 Nil value allowance or restriction is only significant to elements of non-textual types (e.g.,
1480 dates and numerics), and elements of text types that have restricted value space (e.g.,
1481 code). This is because an unrestricted text typed element always contains the empty
1482 string (“”) in its value space. However, for numerics and restricted text type elements,

1483 NIEM allows users to tighten constraints as required in IEPDs by resetting
1484 `nillable="false".`

1485 **5.6.2. Attribute Declarations**

1486 Attribute declarations must be declared with a type by **[Rule 5-10]**, which forbids
1487 anonymous type definitions for attributes.

1488 Within an attribute declaration, the attribute `fixed` may be used as per the XML Schema
1489 specification. Within an attribute declaration, the attribute `form` is irrelevant to NIEM, as
1490 NIEM-conformant schemas may not contain local attribute declarations.

1491 Attribute uses (attribute declarations acting as particles) must be uses of top-level named
1492 attributes. NIEM-conformant schemas may not define local named attributes within type
1493 definitions. Within an attribute use, the attributes `fixed` and `use` may be used as per
1494 the XML Schema specification.

1495 **5.6.3. Attribute Group Definitions**

1496 In NIEM conformant schemas, use of attribute groups is restricted. The only attribute
1497 group that plays a part in NIEM-conformant schemas is
1498 `structures:SimpleObjectAttributeGroup`. This attribute group provides the
1499 attributes necessary for IDs, metadata, and link metadata.

1500 **[Rule 5-57]**

1501 Within a NIEM-conformant schema, any occurrence of the element
1502 `xsd:attributeGroup` MUST own an attribute `ref`.

1503 **[Rationale]**

1504 The only attribute group used in NIEM-conformant schemas is
1505 `structures:simpleObjectAttributeGroup`. Therefore, NIEM-conformant
1506 schemas do not define additional attribute groups.

1507 **[Rule 5-58]**

1508 Within a NIEM-conformant schema, the attribute `ref` owned by any element
1509 `xsd:attributeGroup` MUST have a value of a qualified name (possibly using
1510 the default namespace) that SHALL resolve to the namespace for the NIEM
1511 `structures` namespace and the local name
1512 `SimpleObjectAttributeGroup`.

1513 **[Rationale]**

1514 The only attribute group used within NIEM-conformant schemas is
1515 `structures:SimpleObjectAttributeGroup`. Therefore, within a NIEM
1516 conformant schema, only this attribute group can be referenced.

1517

1518 6. Modeling Rules

1519 NIEM provides a framework for modeling concepts and relationships as XML artifacts.
1520 The data model is implemented via XML Schema. However, XML Schema does not
1521 provide sufficient structure and constraint to enable translating from a conceptual model
1522 to a schema, and then to instances of the concepts. NIEM provides additional support for
1523 modeling concepts as schemas, and provides rules for creating and connecting data that
1524 realizes those concepts.

1525 [Definition: NIEM-conformant schema]

1526 A **NIEM-conformant schema** is an XML document which follows the rules for
1527 NIEM-conformant schemas, as provided by this document. Any schema that
1528 follows all of the rules may be called NIEM-conformant.

1529 Underlying the NIEM data model are two namespaces: the `structures` namespace and
1530 the `appinfo` namespace. These two namespaces provide schema components that
1531 serve two functions:

- 1532 1. They provide support for connecting structural definitions to concepts
- 1533 2. They provide base components from which to derive structural definitions.

1534 These namespaces are distributed with the NIEM data model content, but are not
1535 themselves considered to be content of the data model. They are instead, part of the
1536 structure on which the data model is built.

1537 6.1. `xsd:schema` Document Element Restrictions

1538 [Rule 6-1]

1539 Within a NIEM-conformant schema, the document element `xsd:schema` **MUST**
1540 have application information `appinfo:ConformantIndicator`, with text
1541 content `"true"`.

1542 Rationale

1543 The `appinfo:ConformantIndicator` element is how NIEM-conformant
1544 schemas indicate that they are, in fact NIEM-conformant. Without such an
1545 indicator, conformance would have to be "guessed" by readers and processors.

1546 [Rule 6-2]

1547 Two XML schemas **SHALL** have the same value for attribute `targetNamespace`
1548 carried by the element `xsd:schema` if and only if they represent the same set of
1549 components.

1550 [Rule 6-3]

1551 Two XML Schemas **SHALL** have the same value for attribute
1552 `targetNamespace` carried by the element `xsd:schema`, and different values
1553 for attribute `version` carried by the element `xsd:schema` if and only if they are
1554 different views of the same set of components.

1555 Rationale

1556 These rules embody the basic philosophy behind NIEM's use of namespaced
1557 components: A component is uniquely identified by its class (e.g. element,
1558 attribute, type), its namespace (a URI), and its local name (an unqualified string).
1559 Any two matching component identifiers refer to the same component, even if the
1560 versions of the schemas containing each are different.

1561 6.2. Annotations

1562 NIEM-conformant schemas define data models for the purpose of information exchange.
1563 A major part of defining data models is the proper definition of the contents of the model.
1564 What does a component mean, and what might it contain? How should it be used?
1565 NIEM-conformant schemas contain the invariant part of the definitions for the data model.
1566 The set of definitions includes:

- 1567 1. A text definition of each component. This describes what the component means.
- 1568 2. The structural definition of each component. This is made up of XML Schema
1569 components.

1570 When possible, meaning is expressed via XML Schema mechanisms: type derivation,
1571 element substitution, specific types and structures, as well as names that are trivially
1572 parseable. Beyond that, NIEM-specific syntax must be used, as discussed in this
1573 section.

1574 6.2.1. Human-Readable Documentation

1575 By other rules, a schema component must contain at most one element
1576 `xsd:annotation`. An element `xsd:annotation` in turn must contain at most one
1577 element `xsd:documentation`. The content of the element `xsd:documentation` on a
1578 component is the definition for the component.

1579 [Rule 6-4]

1580 Within a NIEM-conformant schema, any type definition MUST be a documented
1581 component.

1582 [Rule 6-5]

1583 Within a NIEM-conformant schema, any element declaration MUST be a
1584 documented component.

1585 [Rule 6-6]

1586 Within a NIEM-conformant schema, any attribute declaration MUST be a
1587 documented component.

1588 [Rule 6-7]

1589 Within a NIEM-conformant schema, the element `xsd:enumeration` MUST be a
1590 documented component.

1591 [Rule 6-8]

1592 Within a NIEM-conformant schema, the document element `xsd:schema` MUST
1593 be a documented component.

1594 Note that [Rule 4-4] applies [ISO 11179 Part 4] definition rules to documented
1595 components.

1596 [Rule 6-9]

1597 Words or synonyms for the words within a data element definition MAY be reused
1598 as terms in the corresponding component name, if those words do not dilute the
1599 semantics and understanding of, or impart ambiguity to, the entity or concept that
1600 the component represents.

1601 [Rule 6-10]

1602 An object class SHALL have one and only one associated semantic meaning (i.e.
1603 a single word sense.) as described in the definition of the component that
1604 represents that object class.

1605 **[Rule 6-11]**

1606 An object class SHALL NOT be redefined within the definitions of the
1607 components that represent properties or subparts of that entity or class.

1608 **Rationale**

1609 Data definitions should be concise, precise, and unambiguous without
1610 embedding additional definitions of data elements that have already been defined
1611 once elsewhere (such as object classes). **[ISO 11179 Part 4]** says that
1612 definitions should not be nested inside other definitions. Furthermore, a data
1613 dictionary is not a language dictionary. It is acceptable to reuse terms (object
1614 class, property term, and qualifier terms) from a component name within its
1615 corresponding definition to enhance clarity, as long as the requirements and
1616 recommendations of **[ISO 11179 Part 4]** are not violated. This further enhances
1617 brevity and precision.

1618 **[Rule 6-12]**

1619 A NIEM data definition SHALL NOT contain explicit representational or data
1620 typing information such as number characters, type of characters, etc., unless
1621 the very nature of the component can only be described by such information.

1622 **Rationale**

1623 A component definition is intended to describe semantic meaning only, not
1624 representation or structure. How a component with simple content is
1625 represented is indicated through the representation term and further refined
1626 through constraints.

1627 **Example 1**

```
1628 <xsd:element name="AngularMinuteValue" type="nc:AngularMinuteType"  
1629           nillable="true">  
1630   <xsd:annotation>  
1631     <xsd:documentation>  
1632       A value that specifies a minute of a degree. The value comes  
1633       from a restricted range of 0 (inclusive) to 60 (exclusive).  
1634     </xsd:documentation>  
1635   </xsd:annotation>  
1636 </xsd:element>
```

1637 In Example 1 above, the component definition contains representational information
1638 because the component is mathematical and therefore requires such. In Example 2
1639 below, the definition is incorrect and states unnecessary representational information
1640 about the data element. `nc:PersonSSNIdentification` is not a Social Security
1641 Number (SSN); it is a complex element (type `nc:IdentificationType`) that contains
1642 a SSN identifier as well as other properties that describe a person's SSN identifier (such
1643 as issue date, issue authority, etc.). The phrase "9-digit" is incorrect and unnecessary
1644 because it only applies to the SSN identifier and should be applied as a length or pattern
1645 constraint on the identifier only.

1646

Example 21647
1648
1649
1650
1651
1652
1653
1654
1655

```
<xsd:element name="PersonSSNIdentification" type="nc:IdentificationType">
  <xsd:annotation>
    <xsd:documentation>
      A social security number that references a person; a 9-digit
      numeric identifier assigned to a living person by the United
      States Social Security Administration.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

1656

[Rule 6-13]

1657

A component definition SHALL begin with a standard opening phrase that depends on the class of the component per Table 1: Standard Opening Phrases:

1658
1659**Table 1: Standard Opening Phrases**

ThisComponent Class	Definition opening phrase
Abstract	"A data concept for a ..."
Association	"A relationship ..."
Augmentation	"Supplements ..."
Entities and properties of such	"A (An) ..."
Indicator	"True if ...; false otherwise/if..."
Role	"Acts as ..."
Type	"A data type for ..."
Role	"Acts as ..."

1660

Rationale1661
1662
1663
1664

A standard opening phrase base on component class helps to ensure consistent definitions that appropriate for the type of component item being defined. These opening phrases also provide a cue that facilitates recognition of the particular kind of component.

1665

6.2.2. Machine-Readable Annotations1666
1667
1668
1669
1670
1671

XML Schema provides *application information* schema components to provide for automatic processing and machine-readable content for schemas. NIEM utilizes application information to convey information that is outside schema definition, and outside human-readable text definitions. NIEM uses application information to convey high-level data model concepts and additional syntax to support the NIEM conceptual model and validation of NIEM-conformant XML instances.

1672
1673
1674

NIEM defines a single namespace which holds components for use in NIEM-conformant schema application information. This namespace is referred to as the `appinfo` namespace.

1675

[Definition: appinfo namespace]1676
1677

The **appinfo namespace** is the namespace represented by the URI "`http://niem.gov/niem/appinfo/2.0`".

1678
1679

The `appinfo` namespace defines elements which provide additional semantics and syntactic guidelines for components built by NIEM schemas.

1680

[Rule 6-14]

1681

A NIEM-conformant schema SHALL import the `appinfo` namespace.

1682 **Rationale**

1683 For uniformity, all NIEM-conformant schemas must import the appinfo
1684 namespace.

1685 **[Definition: application information]**

1686 A component is said to have **application information** of some element **E** when
1687 the root element that defines the component has an immediate child element
1688 `xsd:annotation`, which has an immediate child element `xsd:appinfo`, which
1689 has as an immediate child the element **E**.

1690 If a component is described as "having application information", this means that the
1691 application information elements under consideration are children of the element which
1692 defines the component.

1693 The majority of uses of application information from the `appinfo` namespace are
1694 described in the modeling rules for the specific component.

1695 **6.2.2.1. Deprecation**

1696 The `appinfo` schema provides a construct for indicating that a construct is deprecated.
1697 A deprecated component is one whose use is not recommended. A deprecated
1698 component is kept in a schema for support of older versions, but should not be used in
1699 new efforts. A deprecated component will be removed, replaced or renamed in a later
1700 edition of a schema.

1701 **[Definition: deprecated component]**

1702 In a particular NIEM-conformant namespace, a **deprecated component** is one
1703 whose use is not recommended, yet which is maintained in the schema for
1704 compatibility with previous versions of the namespace.

1705 **[Rule 6-15]**

1706 A component which is deprecated SHALL be indicated as such by the component
1707 having application information `appinfo:Deprecated`, with an attribute `value`
1708 with a value of `true`.

1709 **Rationale**

1710 Deprecation can allow version management to be more consistent; versions of
1711 schema may be incrementally improved, without introducing validation problems
1712 and incompatibility. As XML Schema lacks a deprecation mechanism, NIEM
1713 defines such a mechanism.

1714 **6.2.2.2. Indicating Conformance**

1715 The element `appinfo:ConformantIndicator` is used for two purposes.

- 1716 1. To indicate that a schema is conformant, or that it represents a conformant
1717 namespace.
- 1718 2. To indicate that an imported schema is not conformant, or represents a non-
1719 conformant namespace.

1720 The specific rules concerning this element appear in Section 6.1, `xsd:schema`
1721 Document Element Restrictions, and Section 6.6, Using External Schemas.

1722 **6.2.2.3. Bases of Derived Components**

1723 The `appinfo` namespace provides an annotation for indicating the base of a derived
1724 component. This is expressed via the `appinfo:Base` application information.

1725 **[Rule 6-16]**

1726 Within a NIEM-conformant schema, the element `appinfo:Base` MAY be used
1727 in one of the following ways:

- 1728 1. By a type definition, to indicate the base type, or `structures:Object`
1729 or `structures:Association`, or
- 1730 2. By an element declaration, to indicate the base element

1731 The element `appinfo:Base` SHALL NOT be used for any other purpose.

1732 **Rationale**

1733 The `appinfo:Base` element is required to clarify semantics of types as object or
1734 association types, when such derivation is not otherwise derivable from the
1735 component definitions.

1736 **[Rule 6-17]**

1737 Within a NIEM-conformant schema, the element `appinfo:Base` SHALL
1738 indicate, by namespace and name, one of the following:

- 1739 1. a NIEM-conformant schema component, or
- 1740 2. `structures:Object`, or
- 1741 3. `structures:Association`.

1742 **[Rule 6-18]**

1743 Within a NIEM-conformation schema, an attribute `appinfo:namespace` owned
1744 by an element `appinfo:Base` SHALL have a value of either:

- 1745 1. a namespace which is the target namespace of a NIEM-conformant
1746 schema, or
- 1747 2. the `structures` namespace.

1748 **[Rule 6-19]**

1749 Within a NIEM-conformant schema, an element `appinfo:Base` which does not
1750 own an attribute `appinfo:namespace` SHALL refer to the target namespace of
1751 the schema in which it is used.

1752 **[Rule 6-20]**

1753 Within a NIEM-conformant schema, an element `appinfo:Base` SHALL own an
1754 attribute `appinfo:name`.

1755 **[Rule 6-21]**

1756 Within a NIEM-conformant schema, if an element `appinfo:Base` indicates a
1757 NIEM-conformant namespace, then the value of the attribute `appinfo:name`
1758 owned by the element `appinfo:Base` SHALL indicate a schema component in
1759 the indicated namespace.

1760 **[Rule 6-22]**

1761 Within a NIEM-conformant schema, if an element `appinfo:Base` indicates the
1762 `structures` namespace, then the value of the attribute `appinfo:name` owned by
1763 the element `appinfo:Base` SHALL have a value of one of:

- 1764 1. `structures:Object`, or
- 1765 2. `structures:Association`, or

1766 3. a schema component defined by the `structures` schema.

1767 **Rationale**

1768 Together, this set of rules establishes the element `appinfo:Base` as a
1769 reference to either a NIEM-conformant schema component, or to a special NIEM
1770 component, which acts as the base for the containing schema component. .

1771 **6.2.2.4. Application of Constructs**

1772 NIEM schemas provide capability for modeling beyond that provided by basic XML
1773 Schema. Two methods made available by NIEM are augmentations and metadata. Both
1774 of these methods create schema components which may be applied to types in specific
1775 ways. The applicability of these components to types is expressed with the
1776 `appinfo:AppliesTo` element.

1777 **[Rule 6-23]**

1778 Within a NIEM-conformant schema, the element `appinfo:AppliesTo` MAY be
1779 used in any of the following ways:

- 1780 1. To indicate a base type to which an augmentation may be applied
- 1781 2. To indicate a base type to which a metadata type may be applied

1782 The element `appinfo:AppliesTo` SHALL NOT be used for any other purpose.

1783 **Rationale**

1784 The `appinfo:AppliesTo` element is required to express constraints beyond
1785 those available within XML Schema. Use of this element allows advanced
1786 processing of instances and schemas for type-safety.

1787 **[Rule 6-24]**

1788 Within a NIEM-conformant schema, the element `appinfo:AppliesTo` SHALL
1789 indicate a schema component, by namespace and name.

1790 **[Rule 6-25]**

1791 Within a NIEM-conformation schema, an attribute `appinfo:namespace` owned
1792 by an element `appinfo:AppliesTo` SHALL indicate the namespace of the
1793 type to which `appinfo:AppliesTo` refers. The indicated namespace SHALL
1794 be NIEM-conformant.

1795 **[Rule 6-26]**

1796 The type to which the attribute `appinfo:appliesTo` refers MUST be the
1797 indicated type or MUST be transitively derived from the indicated type.

1798 **[Rule 6-27]**

1799 Within a NIEM-conformant schema, an element `appinfo:AppliesTo` which
1800 does not carry an attribute `appinfo:namespace` SHALL refer to the target
1801 namespace of the schema in which it is used.

1802 **[Rule 6-28]**

1803 Within a NIEM-conformant schema, an element `appinfo:AppliesTo` SHALL
1804 carry an attribute `appinfo:name`. The value of this attribute SHALL indicate the
1805 local name of a schema component within the namespace specified by the
1806 element.

1807 **Rationale**

1808 Together, this set of rules establishes the element `appinfo:AppliesTo` as a
1809 reference to a NIEM-conformant schema component to which a NIEM construct
1810 may be applied.

1811 **6.2.2.5. Targets of References**

1812 NIEM provides references, in order to avoid problems occurring when only XML element
1813 containment is available. The `appinfo:ReferenceTarget` element specifies the type
1814 to which a reference element may be applied.

1815 **[Rule 6-29]**

1816 Within a NIEM-conformant schema, the element `appinfo:ReferenceTarget`
1817 SHALL specify the type of a schema component which an instance of a reference
1818 element references. The element `appinfo:ReferenceTarget` SHALL NOT
1819 be used for any other purpose.

1820 **[Rule 6-30]**

1821 A reference element SHALL reference an instance of the indicated type, or an
1822 instance of a type derived from that type.

1823 **Rationale**

1824 The element `appinfo:ReferenceTarget` is required to express the type of
1825 referenced content. This level of type-safety is not provided by XML Schema.

1826 **[Rule 6-30.1]**

1827 Within a NIEM-conformant schema, a reference element MUST have at most one
1828 instance of the element `appinfo:ReferenceTarget`.

1829 **Rationale**

1830 Content elements in XML Schema may have at most one type. This rule ensures
1831 that reference elements follow the same pattern.

1832 **[Rule 6-31]**

1833 Within a NIEM-conformant schema, the element `appinfo:ReferenceTarget`
1834 SHALL indicate a type definition schema component, by namespace and name.

1835 **[Rule 6-32]**

1836 Within a NIEM-conformation schema, an attribute `appinfo:namespace` carried
1837 by an element `appinfo:ReferenceTarget` SHALL indicate the namespace of
1838 the referenced schema component. The indicated namespace SHALL be NIEM-
1839 conformant.

1840 **[Rule 6-33]**

1841 Within a NIEM-conformant schema, an element `appinfo:ReferenceTarget`
1842 which does not carry an attribute `appinfo:namespace` SHALL refer to the
1843 target namespace of the schema in which it is used.

1844 **[Rule 6-34]**

1845 Within a NIEM-conformant schema, an element `appinfo:ReferenceTarget`
1846 SHALL carry an attribute `appinfo:name`. The value of this attribute SHALL
1847 indicate the local name of a type definition schema component within the
1848 namespace specified by the element.

1849 **Rationale**

1850 Together, this set of rules establishes the element `appinfo:ReferenceTarget`
1851 as a reference to a NIEM-conformant type definition schema component which a
1852 reference element instance may reference.

1853 **6.3. Complex Type Definitions**

1854 Under XML Schema rules, a CCC (complex type with complex content) may not be the
1855 base type of a CSC (complex type with simple content), and a CSC may not be a base
1856 for a CCC. Therefore, NIEM defines one pattern for defining a CCC, and a different
1857 pattern for defining a CSC. These patterns supply common base definitions that will be
1858 provided for CSCs and CCCs. These patterns are established by the rules for use of
1859 `xsd:extension` in `xsd:complexContent` and `xsd:simpleContent` elements. The
1860 relevant rules may be found in Sections 5.5.3, Simple Content (CSC) Restrictions, and
1861 5.5.4, Complex Content (CCC) Restrictions.

1862 **[Rule 6-35]**

1863 Within a NIEM-conformant schema, a complex type definition SHALL be one of
1864 the following classes of types:

- 1865 1. An object type
- 1866 2. A role type
- 1867 3. An association type
- 1868 4. A metadata type
- 1869 5. An augmentation type
- 1870 6. An adapter type.

1871 **Rationale**

1872 This rule establishes the classes of NIEM complex types. It is a limited set, each
1873 class with distinct semantics.

1874 The first five types are described in subsections below. The adapter type is described in
1875 Section 6.6, Using External Schemas.

1876 **[Rule 6-36]**

1877 Within a NIEM-conformant schema, an element MUST NOT be introduced more
1878 than once into the direct content of a type definition. This applies to content
1879 acquired through extension of base types. This does not apply to a base
1880 element or derived element to one previously existing in the type definition.

1881 **Rationale**

1882 This rule ensures that sequences of elements are simple sequences. A type
1883 should not define, for example, a sequence of elements A, B, then A again.
1884 Definitions should define, instead, what elements may be included, and their
1885 cardinality. Specific orders should be expressed in instances, when necessary,
1886 by the use of the attribute `structures:sequenceID`.

1887 **6.3.1. Object Types**

1888 **[Definition: object type]**

1889 In a NIEM-conformant schema, an **object type** is a complex type definition, an
1890 instance of which asserts the existence of an object. An object type represents
1891 some kind of object: a thing with its own lifespan that has some existence. The
1892 object may or may not be a physical object. It may be a conceptual object.

1893 **[Rule 6-37]**

1894 Within a NIEM-conformant schema, an object type SHALL be a complex type
1895 definition that has one of the following forms:

- 1896 1. Has simple content, is based on a simple type, and contains the attribute
1897 group `structures:SimpleObjectAttributeGroup`, and has
1898 application information `appinfo:Base of structures:Object`, or
- 1899 2. Has complex content, and is based on complex type
1900 `structures:ComplexObjectType`, and has application information
1901 `appinfo:Base of structures:Object`, or
- 1902 3. Is a complex type that is derived from an object type, which is defined
1903 according to this rule.

1904 **Rationale**

1905 Object types are at the core of NIEM. They are built in a uniform way, from a
1906 simple design pattern: they take one of the two "root" forms outlined above, or
1907 they are built from other object types, depending on whether they are of simple or
1908 complex content.

1909 **6.3.2. Role Types**

1910 NIEM differentiates between an object and a role of the object. The term "role" is used
1911 here to mean a function or part played by some object.

1912 **[Definition: role type]**

1913 A **role type** is a type that represents a particular function, purpose, usage, or role
1914 of an object.

1915 The simplest way to represent a role of an object is to use an element. The following
1916 example represents the role of a person who performs an assessment:

1917

```
<xsd:element name="AssessmentPerson" type="nc:PersonType"/>
```

1918 In many cases, there is a further need to represent characteristics and additional
1919 information associated with a role of an object. In such cases, the above element is
1920 insufficient. For example, when a person is a driver involved in a automotive crash, the
1921 person plays the role of a `j:CrashDriver`. In the case of a crash, there is more
1922 information associated with the role of the driver than just his identity for the role. One
1923 such example would be the traffic violation code, `j:CrashDriverViolationCode` is
1924 frequently a characteristic property of a `j:CrashDriver`. For this reason, a role type,
1925 `j:CrashDriverType` is created.

1926 A role type provides the location for information associated with an object playing a role.
1927 A role type is used instead of the base type (in this case, `nc:PersonType`). The role
1928 type holds information specific to the role, but not specific to the context or the base
1929 object (the object that plays the role). Developers of NIEM-conformant schemas should
1930 create and use role types whenever they have non-persistent information specific to a
1931 base object. Such information generally expires when the base object is no longer
1932 playing the role. Information that is persistent to the base object probably does not
1933 belong in a role type.

1934 **[Definition: RoleOf element]**

1935 In a NIEM-conformant schema, a **RoleOf element** is a reference element whose
1936 type is the base type of the role.

1937 Here is an example of a role type from the NIEM Justice domain which uses a RoleOf
1938 element:

```
1939 <xsd:complexType name="CrashPersonType">  
1940   ...  
1941   <xsd:sequence>  
1942     <xsd:element ref="nc:RoleOfPersonReference" minOccurs="0"  
1943       maxOccurs="unbounded"/>  
1944     ...  
1945     <xsd:element ref="j:CrashPersonInjury" minOccurs="0"  
1946       maxOccurs="unbounded"/>  
1947     ...  
1948     <xsd:element ref="j:AlcoholTestResultCode" minOccurs="0"  
1949       maxOccurs="unbounded"/>  
1950     ...  
1951   </xsd:sequence>  
1952   ...  
1953 </xsd:complexType>
```

1954 `nc:RoleOfPersonReference` is defined as “An entity of whom the role object is a
1955 function.” In this example, the role object is `j:CrashPersonType` and the base type of
1956 the role object is a `nc:PersonType`, the entity of whom `j:CrashPersonType` is a
1957 function (per the definition above).

1958 This role object represents a particular role of a person: a person involved in a vehicular
1959 crash. It refers to the person of whom this object is a role through the
1960 `nc:RoleOfPersonReference` element. It also includes additional information
1961 particular to the person's role in the crash.

1962 [Rule 6-38]

1963 Within a NIEM-conformant schema, any element with a name beginning with the
1964 string `RoleOf` SHALL represent a base type, of which the containing type
1965 represents a role.

1966 Rationale

1967 A “RoleOf” element references its corresponding base element. The “RoleOf”
1968 label on the reference element ensures that a role object is distinguishable from
1969 other objects and its link to the associated base is also distinguishable from the
1970 additional properties that are characteristic of this role or that add information.

1971 NIEM does not require that there be only one RoleOf element within a single type.
1972 However, the use of multiple RoleOf elements may not make sense, and indeed, an
1973 example of a role that references two or more base types is very difficult (if not
1974 impossible) to conceive.

1975 An object should be a role of only a single object. However, there may be varied
1976 assertions of what object that might be, or time constraints on the role. Many exchanges
1977 may wish to restrict RoleOf elements to a single occurrence within a type.

1978 Role elements are generally reference elements, targeting the base type. That is, a role
1979 element is usually a reference element, not a content element.

1980 6.3.3. Association Types

1981 Within NIEM, an association is a specific relationship between objects. Associations are
1982 used when a simple NIEM property is insufficient to model the relationship clearly and
1983 when properties of the relationship exist that are not attributable to the objects being
1984 related.

1985 **[Definition: association type]**
1986 In a NIEM-conformant schema, an **association type** is a type which establishes
1987 a relationship between objects, along with the properties of that relationship. An
1988 association type provides a structure which does not establish existence of an
1989 object, but instead specifies relationships between objects.

1990 **[Definition: association]**

1991 In a NIEM-conformant schema, an **association** is an element whose type is a
1992 association type.

1993 **[Rule 6-39]**

1994 Within a NIEM-conformant schema, an association type SHALL be a complex
1995 type definition that has one of the following forms:

- 1996 1. Has complex content, is based on the complex type
1997 `structures:ComplexObjectType`, and has application information
1998 `appinfo:Base of structures:Association`, or
1999 2. Is a complex type that is derived from an association type, which is
2000 defined according to this rule.

2001 **Rationale**

2002 Associations are easily identifiable as such, and have a commonly-defined base
2003 type.

2004 **[Rule 6-40]**

2005 Within a NIEM-conformant schema, in an association type, any element which
2006 represents a participant in the relationship established by the association type
2007 SHALL be a reference element.

2008 **Rationale**

2009 Associations are intended to relate objects defined elsewhere. They are not
2010 intended to carry content of participant objects.

2011 **6.3.4. Metadata Types**

2012 Within NIEM, metadata is defined as “data about data.” This may include information
2013 such as the security of a piece of data, or source of the data. These pieces of metadata
2014 may be composed into a metadata type. The types of data to which metadata may be
2015 applied may be constrained.

2016 **[Definition: metadata type]**

2017 A **metadata type** describes data about data, that is, information which is not
2018 descriptive of objects and their relationships, but is descriptive of the data itself.
2019 It is useful to provide a general mechanism for data about data. This provides
2020 required flexibility to precisely represent information.

2021 **[Definition: metadata element]**

2022 Within a NIEM-conformant schema, a **metadata element** is an element whose
2023 type is a metadata type. There are specific limitations on the meaning of a
2024 metadata element in an instance; it does not establish existence of an object, nor
2025 is it a property of its containing object.

2026 **[Rule 6-41]**

2027 Within a NIEM-conformant schema, a metadata type SHALL contain elements
2028 appropriate for a specific class of data about data.

2029 **[Rule 6-42]**

2030 Within a NIEM-conformant schema, a metadata type and only a metadata type
2031 SHALL be derived directly from `structures:MetadataType`.

2032 **Rationale**

2033 A metadata type establishes a specific, named aggregation of data about data.
2034 Any type derived from `structures:MetadataType` is a metadata type.
2035 Metadata types should not be derived from other metadata types. Such
2036 metadata types should be used as-is, and additional metadata types defined for
2037 additional content.

2038 **[Rule 6-43]**

2039 Within a NIEM-conformant schema, a metadata type MAY have application
2040 information `appinfo:AppliesTo`, indicating the NIEM-conformant object,
2041 association, or external adapter types to which the metadata applies.

2042 **[Rule 6-44]**

2043 Within a NIEM-conformant schema, a metadata type which does not have
2044 application information `appinfo:AppliesTo` MAY be applied to any object
2045 type, association type, or external adapter type.

2046 **Rationale**

2047 Metadata may be constrained to be applicable to only specific types, or it may be
2048 defined to be applicable to any type. Information such as the source of a piece of
2049 data, or the security classification of a piece of data are examples of metadata
2050 that may be considered globally applicable.

2051 **6.3.5. Augmentation Types**

2052 Builders of domains and extensions to NIEM distribution schemas need to be able to
2053 define extensions to types. However, extension of types by multiple domain schemas
2054 and extension schemas proves problematic, as it results in multiple extensions of a single
2055 type. XML Schema does not provide for multiple types of an instance, and so such a
2056 method results in duplication of base type content, and a need to resolve "same-as"
2057 relationships between the instances of the various derived types.

2058 Instead, it is preferable for domains and extensions to provide augmentations. These are
2059 reusable types, and elements of those types, which may be added to an object class, in a
2060 single extended type, by the author of a NIEM-conformant schema. This avoids the
2061 problem of multiple extended types, but allows domains and extensions to define
2062 reusable extensions.

2063 Augmentation types such as `dom:PersonAugmentationType` (where `dom:` is a NIEM
2064 domain namespace) exist to extend NIEM Core types such as `nc:PersonType` without
2065 creating a new specialized object within the model. Augmentation types are never
2066 applied within the model to the types they are designed to augment. Doing so would
2067 restrict reusing and combining these augmentations.

2068 Instead, augmentation should be applied within IEPDs. So, in an IEPD (NOT within
2069 NIEM), base `nc:PersonType` may be extended, for example, as `my-`
2070 `iepd:PersonType` by adding elements `a:PersonAugmentation` and
2071 `b:PersonAugmentation`. As a result, `my-iepd:PersonType` will contain all the
2072 properties in `nc:PersonType` plus the properties in both of the elements
2073 `a:PersonAugmentation` and `b:PersonAugmentation`, which, in turn, each contain
2074 their respective sets of sub-elements.

2075 All NIEM augmentation types extend the abstract type
2076 `structures:AugmentationType`. Therefore, all augmentation types automatically
2077 contain the attributes `structures:id` and `structures:metadata` for referencing
2078 and metadata respectively. NIEM also provides the abstract element
2079 `structures:Augmentation` (of type `structures:AugmentationType`) as the
2080 common substitution group head for all augmentation elements. An augmentation
2081 element placed into this substitution group can be used in an instance wherever
2082 `structures:Augmentation` occurs in the corresponding IEPD schema. The user
2083 must follow NIEM naming conventions for augmentation component names, and must
2084 place new augmentation elements into the `structures:Augmentation` substitution
2085 group. Furthermore, if an augmentation element cannot be applied to all types in the
2086 model, then the user must document those types that the new augmentation element can
2087 be applied to using the `appinfo:AppliesTo` element.

2088 **[Definition: augmentation type]**

2089 An **augmentation type** is a complex type which provides a reusable block of
2090 data which may be added to object types or association types.

2091 **[Definition: augmentation]**

2092 An **augmentation** of a NIEM-conformant object type is a block of additional data
2093 added to an object type, in order to carry additional data beyond that of the
2094 original object definition.

2095 **[Rule 6-45]**

2096 An augmentation type:

- 2097 1. SHALL be transitively derived from `structures:AugmentationType`
2098 and
- 2099 2. SHALL contain elements which represent properties to be applied to a
2100 base type.

2101 **Rationale**

2102 A base type is the type to which an augmentation is to be applied. An
2103 augmentation may be applied to any number of types. Base types are assigned
2104 by augmentation elements.

2105 **[Rule 6-46]**

2106 Within a NIEM-conformant schema, an augmentation element definition:

- 2107 1. SHALL have a type which is an augmentation type
- 2108 2. SHALL use the `substitutionGroup` attribute such that it is transitively
2109 substitutable for the element `structures:Augmentation`

2110 An element which is not an augmentation element SHALL NOT meet either of the
2111 above criteria.

2112 **Rationale**

2113 An augmentation is trivially identifiable as such. The use of the common
2114 `structures:Augmentation` element allows message builders to optionally
2115 delay specifying augmentations to be applied to a type until runtime.

2116 **[Rule 6-47]**

2117 Within a NIEM-conformant schema, an element definition for an augmentation
2118 element MAY contain one or more instances of the element

2119 `structures:AppliesTo` as application information, to specify types to which
2120 the augmentation element applies.

2121 **[Rule 6-48]**

2122 Within a NIEM-conformant schema, an element definition for an augmentation
2123 element which does not contain any instances of the element
2124 `structures:AppliesTo` MAY be applied to any object or association type.

2125 **Rationale**

2126 These rules allow schema builders to establish applicability for augmentations.
2127 An augmentation may be applicable to specific types.

2128 Users who wish to apply an augmentation type to a given object type may do so
2129 by creating a new augmentation element, applicable to the object type.

2130 **6.4. Component Usage**

2131 **[Rule 6-49]**

2132 Any type definition referenced by a component within a NIEM-conformant
2133 schema MUST be from one of the following:

- 2134 1. The schema being defined
- 2135 2. A namespace imported as NIEM-conformant
- 2136 3. The XML Schema namespace
- 2137 4. The `structures` namespace.

2138 **Rationale**

2139 NIEM-conformant schemas are based on other NIEM-conformant schemas, and
2140 the supporting namespaces. This simplifies processing and understanding of
2141 data.

2142 **[Rule 6-50]**

2143 Any element declaration referenced by a component within a NIEM-conformant
2144 schema MUST be from one of the following:

- 2145 1. The schema being defined
- 2146 2. A namespace imported as NIEM-conformant
- 2147 3. The `structures` namespace
- 2148 4. An external namespace, in accordance with the rules for external
2149 schemas as specified by this specification.

2150 **[Rule 6-51]**

2151 Any attribute declaration referenced by a component within a NIEM-conformant
2152 schema MUST be from one of the following:

- 2153 1. The schema being defined
- 2154 2. A namespace imported as NIEM-conformant
- 2155 3. The `structures` namespace
- 2156 4. The XML namespace
- 2157 5. An external namespace, in accordance with the rules for external
2158 schemas as specified by this specification.

2159 **Rationale**

2160 NIEM-conformant schemas are based on other NIEM-conformant schemas. All
2161 attributes and elements must be from NIEM-conformant schemas, the
2162 `structures` namespace, the XML namespace, or an external namespace. This
2163 applies to elements referenced for substitution groups, as well. It does not apply
2164 to content of the schema (e.g. within annotations), or to the XML Schema
2165 declarations themselves. It applies only to attributes and elements referenced by
2166 the XML Schema components.

2167 **6.5. NIEM Structural Facilities**

2168 NIEM provides the structures schema which contains base types for types defined in
2169 NIEM-conformant schemas. It provides base elements to act as heads for substitution
2170 groups. It also provides attributes that provide facilities not otherwise provided by XML
2171 Schema. These structures should be used to augment XML data. The structures
2172 provided are not meant to replace fundamental XML organization methods; they are
2173 intended to assist them.

2174 **[Definition: structures namespace]**

2175 The **structures namespace** is the namespace represented by the URI
2176 "<http://niem.gov/niem/structures/2.0>".

2177 The structures namespace is a single namespace, separate from namespaces that define
2178 NIEM-conformant data. This document refers to this content via the prefix `structures`.

2179 **[Rule 6-52]**

2180 A NIEM-conformant schema **MUST** import the NIEM `structures` namespace.

2181 **Rationale**

2182 For uniformity, all NIEM-conformant schemas must import the `structures`
2183 namespace.

2184 **[Rule 6-53]**

2185 NIEM-conformant schemas and instances **MUST** use content within the NIEM
2186 structures namespace as specified in this document and **ONLY** as specified by
2187 this document.

2188 **Rationale**

2189 This rule further enforces uniformity and consistency by mandating use of the
2190 NIEM structures namespace as is, without modification. Users are not allowed to
2191 insert types, attributes, etc. that are not specified by this document (the NDR).

2192 **6.5.1. Sequence ID**

2193 NIEM provides the attribute `structures:sequenceID` for specification of sequential
2194 order of instances, when a complex type's defined element sequence is insufficient. A
2195 limitation of XML Schema is that control of cardinality (the number of times an element
2196 may occur in an instance) requires the use of sequences of elements. This use of
2197 `xsd:sequence` defines the elements occurring within a type in a specific order. This
2198 order may not match the desired sequential order of the represented entities.

2199 An example would be for proper names, where the natural order of the names may not
2200 appear in the same order as the sequence defined by a complex type. Consider the
2201 example:

- 2202 • One address represents the postal code before the city name
- 2203 • Another address represents the city name before the postal code

2204 • The address structure must be defined in exactly one way

2205 Without the `structures:sequenceID` attribute, this example would create a dilemma:
 2206 which address to represent properly, and which to represent incorrectly? The
 2207 `structures:sequenceID` attribute allows the schema sequence to be separated from
 2208 the implied meaning.

2209 As another example, when using a derived type, within an instance, the base type's
 2210 elements occur first, followed by any elements added by extension. If those elements
 2211 need to be interleaved into the existing structure for the proper meaning to be conveyed,
 2212 the `structures:sequenceID` attribute is called for.

2213 The `structures:sequenceID` attribute allows instances to express the sequential
 2214 order of data relative to a parent. The order of data is as yielded by XSLT's `xsl:sort`
 2215 element, with data-type of `xsl:number`, and order of `ascending`. Content with
 2216 identical `structures:sequenceID` values has undefined order.

2217 **[Rule 6-54]**

2218 Within a NIEM-conformant schema, a complex type definition SHALL include the
 2219 attribute `structures:sequenceID` if the order of an occurrence of the type,
 2220 within its parent, relative to its siblings, is meaningful and pertinent, and if the
 2221 content presented by all instances defined by the schema will not otherwise
 2222 occur in the desired sequential order.

2223 **Rationale**

2224 This rule indicates that, if order is meaningful, and the schema won't always
 2225 represent the desired order, then data modelers need to include `sequenceID` to
 2226 allow the proper order to be represented in instances.

2227 Use of `sequenceID` is restricted by are found in the rules on conformant instances in
 2228 Section 7.4, Component Ordering.

2229 **6.5.2. Reference Elements**

2230 In XML instances, relationships between data objects are expressed as XML elements:

2231 1. Data objects are expressed as XML elements, and
 2232 2. XML elements contain attributes and other elements.

2233 In this way, there is generally some implicit relationship between the outer element (the
 2234 "containing" element, a.k.a. the parent element) and the inner elements (the "contained"
 2235 elements, a.k.a. the child elements). Such expression of relationships is said to be by
 2236 containment.

2237 Expression of all relationships via element containment is not always possible. Situations
 2238 that cause problems include:

2239 • Circular relationships. For example, suppose Object1 has a relationship to
 2240 Object2 and Object2 has a relationship to Object1. Expressed via containment,
 2241 this relationship would result in infinite recursive descent.

2242 • Repeated relationships. For example, suppose Object1 has a relationship to
 2243 Object2 and Object3 has a relationship to Object2. Expressed via containment,
 2244 this would result in a duplicate of Object2.

2245 A method that solves this problem is to use references. In a C or assembler, a pointer
 2246 would be used. In C++, a reference might be used. In Java, a reference value might be
 2247 used. The method defined by the XML standard is the use of `ID` and `IDREF`. An `ID`
 2248 refers to an `IDREF`. NIEM uses this method, and assigns to it specific semantics.

2249 **[Definition: reference element]**

2250 A **reference element** is an element that refers to its value by a reference

2251 attribute, instead of carrying it as content.

2252 **[Rule 6-55]**

2253 Within a NIEM-conformant schema, a reference element and only a reference

2254 element SHALL be defined to be of type `structures:ReferenceType`.

2255 **Rationale**

2256 Reference elements must be of the reference type, and elements of the

2257 reference type must be reference elements. This rule ensures that users always

2258 create reference elements using `structures:ReferenceType`, and cannot

2259 use `structures:ReferenceType` for any other purpose.

2260 **[Rule 6-56]**

2261 Within a NIEM-conformant schema, a complex type SHALL NOT be defined such

2262 that an instance of that type owns the attribute `structures:ref`.

2263 **Rationale**

2264 The use of references is limited to reference elements. This constrains the

2265 semantics and syntax of references within NIEM instances. Only

2266 `structures:ReferenceType` may use `structures:ref`, which is the only

2267 means for referencing within NIEM-conformant instances.

2268 **[Rule 6-57]**

2269 Within a NIEM-conformant schema, any two elements of the form

2270 *NCName*

2271 and

2272 *NCNameReference*

2273 where the string value of *NCName* is the same in both forms, SHALL be defined

2274 to have identical semantics. The NIEM recognizes no difference in meaning

2275 between a reference element and an element that is not a reference element.

2276 **Rationale**

2277 NIEM-conformant data instances may use concrete data elements and reference

2278 elements as needed, to represent the meaning of the fundamental data. There is

2279 no difference in meaning between reference or concrete data representations.

2280 The two different methods are available for ease of representation. No difference

2281 in meaning should be implied by the use of one method or the other.

2282 Assertions that indicate "included" data is intrinsic, while referenced data is

2283 extrinsic are not valid and are not applicable to NIEM-conformant data instances

2284 and data definitions.

2285 **[Rule 6-58]**

2286 Within a NIEM-conformant schema, if both elements *NCName* and

2287 *NCNameReference* exist, then the `appinfo:ReferenceTarget` of any

2288 *NCNameReference* element MUST be the type of the element *NCName*.

2289 **Rationale**

2290 By **[Rule 6-57]**, any such pair of elements, *NCName* and *NCNameReference*,

2291 will have identical semantics. This rule ensures that a *NCNameReference*

2292 element is documented to refer to the appropriate type (the type of the
2293 corresponding *NCName* element) and no other.

2294 The NIEM structures schema defines `structures:ReferenceType` to require the use
2295 of an attribute `structures:ref`, which is of type `IDREF` as specified by
2296 **[XMLSchemaStructures]**. According to the rules of XML, such an attribute must contain
2297 a value that is represented by an attribute of type `ID`. In NIEM-conformant instance, the
2298 targets of `IDREFs` are expected to be values of the attribute `structures:id`.

2299 The NIEM structures schema defines `structures:ReferenceType` such that it is
2300 unavailable as a base for extension or restriction.

2301 The NIEM structures schema defines `structures:ReferenceType` such that it has an
2302 optional attribute `structures:id`. This may be used to describe additional metadata or
2303 information about the relationship described by an element of type
2304 `structures:ReferenceType`.

2305 Within a NIEM-conformant instance, the element referenced by an attribute
2306 `structures:ref` must be of a type valid for the object of the fundamental element of
2307 the reference element. The attribute `structures:ref` is discussed in more detail in
2308 Section 7.3.

2309 **6.6. Using External Schemas**

2310 There are a variety of commonly-used standards that are represented in XML Schema.
2311 Such schemas are generally not NIEM-conformant. NIEM-conformant schemas may
2312 reference components defined by these external schemas. NIEM-conformant
2313 components may be constructed from non-NIEM schema components.

2314 **[Definition: external schema]**

2315 An **external schema** is any non-supporting schema that is not NIEM-conformant.

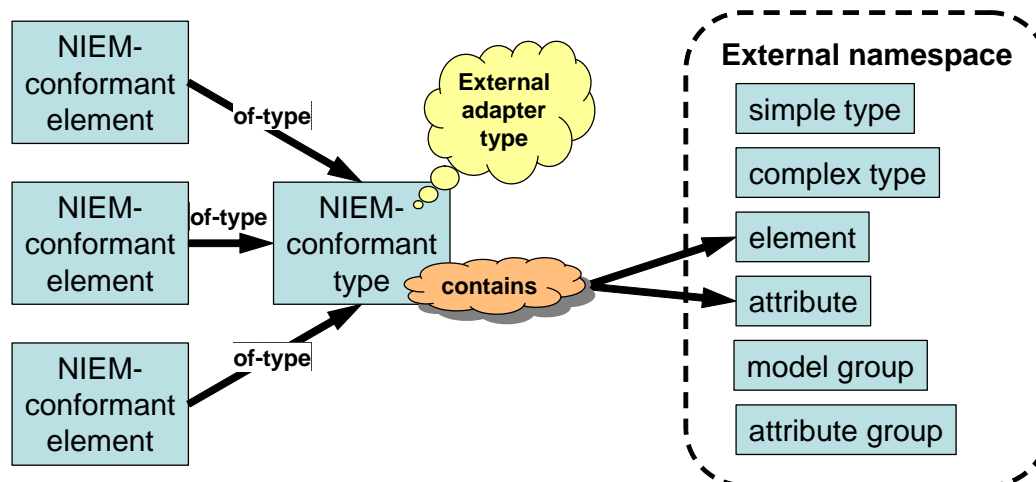
2316 Note that the supporting schemas `structures` and `appinfo` are non-conformant
2317 because they define the fundamental framework on which NIEM is built. However, they
2318 are not considered external schemas because of their supporting nature, and are thus
2319 excluded from this definition.

2320 NIEM-conformant schemas may work with external schemas by creating external adapter
2321 types.

2322 A single method is used to integrate external components into NIEM-conformant
2323 schemas: NIEM-conformant types are constructed from the external components.

2324

Use of external components to create a NIEM-conformant type



2325

2326 Components defined by external schemas are called *external components*. External
 2327 components may be used by a NIEM-conformant type in a specific way: to construct a
 2328 NIEM-conformant type from external components. The goal in this method is to preserve
 2329 as a single unit a set of data that embodies a single *concept* from an external standard.

2330 For example, a NIEM-conformant type may be created to represent a bibliographic
 2331 reference from an external standard. Such an object may be composed of multiple
 2332 elements and types from the external standard. These pieces are put together to form a
 2333 single NIEM-conformant type. For example, an element representing an author, a book,
 2334 and a publisher may be included in a single bibliographic entry.

2335 A NIEM-conformant type built from these components may be used as any other NIEM-
 2336 conformant type. That is, elements may be constructed from such a type, and those
 2337 elements are fully NIEM-conformant.

2338 To construct such a component, a NIEM-conformant schema must first import an external
 2339 schema.

2340 [Rule 6-59]

2341 Within a NIEM-conformant schema, an element `xsd:import` that imports a
 2342 namespace defined by an external schema **MUST** have the application
 2343 information `appinfo:ConformantIndicator`, with a value of `false`.

2344 Rationale

2345 Knowledge of the conformance of an imported schema allows processors to
 2346 understand the semantics of referenced components, without additional
 2347 processing. Namespaces imported into NIEM-conformant schemas are
 2348 assumed to be conformant, unless otherwise indicated.

2349 [Rule 6-60]

2350 Within a NIEM-conformant schema, an element `xsd:import` that imports a
 2351 namespace defined by an external schema **MUST** be a documented component.

2352 Rationale

2353 A NIEM-conformant schema has well-known documentation points. Therefore, a
 2354 schema that imports a NIEM-conformant namespace need not provide additional
 2355 documentation. However, when an external schema is imported, appropriate
 2356 documentation must be provided at the point of import, because documentation

2357 associated with external schemas is undefined and variable. In this particular
 2358 case, documentation of external schemas is required at their point of use in
 2359 NIEM.

2360 **[Definition: adapter type]**

2361 An **adapter type** is a NIEM-conformant type that adapts external components for
 2362 use within NIEM. An adapter type creates a new class of object that embodies a
 2363 single concept composed of external components. An adapter type is defined by
 2364 a NIEM-conformant schema.

2365 **[Rule 6-61]**

2366 Within a NIEM-conformant schema, an adapter type **MUST** have application
 2367 information `appinfo:ExternalAdapterTypeIndicator` with a value of
 2368 `true`. A type that is not an adapter type **SHALL NOT** contain that indicator.

2369 **Rationale**

2370 This rule flags as external adapters those types which may contain external
 2371 content. This allows for easier processing.

2372 **[Rule 6-62]**

2373 Within a NIEM-conformant schema, an adapter type **MUST** be a immediate
 2374 extension of type `structures:ComplexObjectType`.

2375 **Rationale**

2376 The adapter type must contain the content defined for any NIEM component.
 2377 Such content is provided by the complex object type from the `structures`
 2378 namespace.

2379 **[Rule 6-63]**

2380 Within a NIEM-conformant schema, an adapter type **MUST** be composed of only
 2381 elements and attributes from an external standard.

2382 **Rationale**

2383 An adapter type should contain the information from an external standard to
 2384 express a complete concept. This expression should be composed of content
 2385 entirely from an external schema. Most likely, the external schema will be based
 2386 on an external standard, with its own legacy support.

2387 In the case of an external expression that is in the form of model groups, attribute groups,
 2388 or types, additional elements and type components may be created in an external
 2389 schema, and those components may be used by the adapter type.

2390 **[Rule 6-64]**

2391 Within a NIEM-conformant schema, an element reference used in an adapter
 2392 type definition **MUST** be a documented component.

2393 **[Rule 6-65]**

2394 Within a NIEM-conformant schema, an attribute reference used in an adapter
 2395 type definition **MUST** be a documented component.

2396 **Rationale**

2397 In normal (conformant) type definition, a reference to an attribute or element is a
 2398 reference to a documented component. Within an adapter type, the references
 2399 to the attributes and elements being adapted are references to undocumented
 2400 components. These components must be documented to provide
 2401 comprehensibility and interoperability. Since documentation made available by

2402 non-conformant schemas is undefined and variable, documentation of these
2403 components is required at their point of use, within the conformant schema.

2404 **[Rule 6-66]**

2405 Within a NIEM-conformant schema, an adapter type **MUST NOT** be extended or
2406 restricted.

2407 **Rationale**

2408 Adapter types are meant to stand alone; each type expresses a single concept
2409 from an external schema, and adapter types are maintained in separate schemas
2410 which only contain adapter types. In this way, processors may easily switch
2411 modes, processing NIEM-conformant content in one way, and external content in
2412 another.

2413 **6.7. Container Elements**

2414 All NIEM properties establish a relationship between the object holding the property and
2415 the value of the property. For example, an activity object of type `nc:ActivityType`
2416 may have an element `nc:ActivityDescriptionText`. This element will be of type
2417 `nc:TextType` and represents a NIEM property owned by that activity object. An
2418 occurrence of this element within an activity object establishes a relationship between the
2419 activity object and the text: the text is the description of the activity.

2420 In a NIEM-conformant instance, an element establishes a relationship between the object
2421 that contains it and the element's value. This relationship between the object and the
2422 element may be semantically-strong, such as the text description of an activity in the
2423 previous example, or it may be semantically-weak, with its exact meaning left unstated.
2424 In NIEM, the contained element involved in a weakly-defined semantic relationship is
2425 commonly referred to as a **container element**.

2426 A container element establishes a weakly-defined relationship with its containing element.
2427 For example, an object of type `nc:ItemDispositionType` may have a container
2428 element `nc:Item` of type `nc:ItemType`. The container element `nc:Item` does not
2429 establish what relationship exists between the object of `nc:ItemDispositionType`
2430 and itself. There could be any of a number of possible semantics between an object and
2431 the value of a container element - It could be a contained object, a subpart, a
2432 characteristic, or some other relationship. The appearance of this container element
2433 inside the `nc:ItemDispositionType` merely establishes that the disposition has an
2434 item.

2435 The name of the container element is usually based on the NIEM type that defines it:
2436 `nc:PersonType` uses a container element `nc:Person`, while `nc:ActivityType` uses
2437 a container element `nc:Activity`. The concept of an element as a container element
2438 is a notional one.

2439 There are no formalized rules addressing what makes up a container element. A
2440 container element is vaguely defined, and carries very little semantics about its context
2441 and its contents. Accordingly, there is no formal definition of container elements in NIEM:
2442 There are no specific artifacts which define a container element; there are no `appinfo` or
2443 other labels for container elements.

2444 The appearance of a container element within a NIEM type carries no additional
2445 semantics about the relationship between the property and the containing type. Use of
2446 container elements indicate only that there is a relationship, but does not provide any
2447 semantics for interpreting that relationship.

2448 For example, a NIEM container element `nc:Person` would be associated with the NIEM
2449 type `nc:PersonType`. The use of the NIEM container element `nc:Person` in a

2450 containing NIEM type indicates that a person has some association with the instances of
2451 the containing NIEM type. But because the `nc:Person` container element is used, there
2452 is no additional meaning about the association of the person and the instance containing
2453 it. While there is a person associated with the instance, nothing is known about the
2454 relationship except its existence.

2455 The use of the Person container element is in contrast to a NIEM property named
2456 `nc:AssessmentPerson`, also of NIEM type `nc:PersonType`. When the NIEM
2457 property `nc:AssessmentPerson` is contained within an instance of a NIEM type, it is
2458 clear that the person referenced by this property was responsible for an assessment of
2459 some type, relevant to the exchange being modeled. The more descriptive name,
2460 `nc:AssessmentPerson`, gives more information about the relationship of the person
2461 with the containing instance, as compared to the semantic-free implications associated
2462 with use of the `nc:Person` container element.

2463 When a NIEM-conformant schema requires a new container element, it may define a new
2464 element with a concrete type and a general name, with general semantics. Any schema
2465 may define a container element when it requires one. NIEM-conformant schemas may
2466 also create reference elements with general semantics. For example, an element
2467 `nc:PersonReference` will carry the same general, container-like meaning as an
2468 element `nc:Person`.

2469

2470 7. XML Instance Rules

2471 This specification attempts to restrict XML instance data as little as possible, while still
2472 maintaining interoperability.

2473 [Definition: NIEM-conformant document]

2474 A **NIEM-conformant document** is an XML information set whose document
2475 element is defined by a NIEM-conformant schema, and which follows the rules
2476 for conformant element information items as specified by this document.

2477 The terms "XML information set", "document element", and "element information
2478 item" come from [XMLInfoSet]. This definition says that any XML instance
2479 whose document element is a conformant element instance is a NIEM-
2480 conformant document. The word *document* is meant only as used in
2481 [XMLInfoSet].

2482 [Definition: NIEM-conformant element instance]

2483 A **NIEM-conformant element instance** is an XML information item which is
2484 defined by a NIEM-conformant schema, and which follows the rules for
2485 conformant instance data as specified by this document.

2486 XML data may be referred to as a NIEM-conformant instance if it conforms to this
2487 specification.

2488 The NIEM does not require a specific encoding, or specific requirements for the XML
2489 prologue, except as specified by [XML].

2490 7.1. Instance Validation

2491 [Rule 7-1]

2492 A NIEM-conformant instance **MUST** validate to an authoritative NIEM-conformant
2493 schema set for namespaces contained in the instance, and for additional
2494 namespaces required for validation.

2495 Rationale

2496 The schemas which define the exchange must be authoritative. That is, they
2497 must be the reference schema for the namespaces concerned. Other schemas
2498 may be used by application developers for various purposes, but for the
2499 purposes of determining conformance, the authoritative schemas are relevant.

2500 NIEM embraces the use of XML schema instance attributes, including `xsi:type`,
2501 `xsi:nil`, and `xsi:schemaLocation`, as specified by [XMLSchemaStructures].

2502 7.2. Instance Meaning

2503 [Rule 7-2]

2504 Within a NIEM-conformant instance, the meaning of an element with no content
2505 is that additional properties are not asserted. There **SHALL NOT** be additional
2506 meaning interpreted for an element with no content.

2507 Rationale

2508 Elements without content only show a lack of asserted information. That is, data
2509 which is not there is not stated. It may be due to lack of availability, lack of
2510 knowledge, or deliberate withholding of information. If expression of such cases
2511 is required, it should be modeled explicitly.

2512 7.3. Component Representation

2513 NIEM uses element containment for the majority of its data representation needs. That
2514 is, an element containing another element. In general, one object (the content of the
2515 outer element) has a relationship (defined by the name of the inner element) to another
2516 object (the content of the inner element).

2517 Example of element containment

```
2518 <OuterElement>  
2519   <!-- object1: the content of outer element -->  
2520   <InnerElement>  
2521     <!-- object2: the content of inner element -->  
2522   </InnerElement>  
2523   <!-- object1, continued -->  
2524 </OuterElement>
```

2525 This use of the element containment method has limitations. Specifically, recursive and
2526 symmetric relationships (direct or transitive) create difficulties, such as repetition of data,
2527 and resolution of duplicates.

2528 To avoid these problems, NIEM allows references between elements. In this way, one
2529 object (the content of one element) has a relationship (defined by the name of the inner
2530 element) to another object (the content of an element referenced by an attribute of the
2531 inner element).

2532 Example of element reference

```
2533 <OuterElement>  
2534   <!-- object1: the content of outer element -->  
2535   <InnerElementReference structures:ref="object2"/>  
2536   <!-- object1, continued -->  
2537 </OuterElement>  
  
2538 <OtherElement structures:id="object2">  
2539   <!-- object2: the content of other element -->  
2540 </OtherElement>
```

2542 [Rule 7-3]

2543 Within a NIEM-conformant element instance, there SHALL NOT be any
2544 difference in meaning between a property asserted via element containment and
2545 a property asserted by element reference, except as explicitly described by the
2546 semantics of the elements involved.

2547 Rationale

2548 There is no difference in meaning between relationships established by
2549 containment, and those established by reference. They are simply two
2550 mechanisms for expressing connections between objects. Neither mechanism
2551 implies that properties are intrinsic or extrinsic. Such characteristics must be
2552 explicitly stated in property definitions.

2553 Being of type `xsd:ID` and `xsd:IDREF`, validating schema parsers will perform certain
2554 checks on the values of `structures:id` and `structures:ref`. Specifically, no two
2555 IDs may have the same value. This includes `structures:id` and other IDs that may
2556 be used in an instance. Also, any value of `structures:ref` must also appear as the
2557 value of an ID.

2558 **[Rule 7-4]**

2559 Any attribute `structures:ref` MUST have a value which occurs as the value
2560 of an attribute `structures:id` within the same information set.

2561 **Rationale**

2562 This states that in NIEM-conformant content, `structures:ref` attributes must
2563 refer to `structures:id` attributes. This rule ensures that the target of a
2564 reference exists within the same XML instance.

2565 Reference element definitions may include constraints on the type of object which may
2566 be referenced by that element.

2567 **[Rule 7-5]**

2568 Within a NIEM-conformant element instance, given that a reference element is
2569 restricted to a set S of target types T_i , $S = \{ T_1, T_2, \dots, T_n \}$, any attribute
2570 `structures:ref` MUST indicate the value of an attribute `structures:id`
2571 which is owned by an element of a type T such that T is, or is derived from, some
2572 type T_i in S.

2573 **Rationale**

2574 This rule says that the type of the object pointed to by an `structures:ref`
2575 attribute must be of a type specified by the reference element definition. The
2576 restriction of types is defined in the application information of the reference
2577 element definition by the use of the `appinfo:ReferenceTarget` attribute.

2578 **7.4. Component Ordering**

2579 An instance may express the natural order of components by using the order of content
2580 within an XML file. It may also use the `structures:sequenceID` to indicate the order
2581 of components.

2582 **[Rule 7-6]**

2583 The order of elements that are children of a NIEM-conformant element SHALL be
2584 presented as if their sequential order is as follows:

- 2585 1. First, elements owning an attribute `structures:sequenceID`, in the
2586 order that would be yielded with their sequence IDs sorted via XSLT's
2587 `sort` element, with a data type of `number` and an order of `ascending`.
- 2588 2. Following those elements, the remaining elements, in the order in which
2589 they occur within the XML instance.

2590 **Rationale**

2591 Because of NIEM's use of structured, defined types, and its use of
2592 `xsd:sequence`, as well as various representation mechanisms, the order of
2593 data within an XML instance may require more precise definition, and may vary
2594 from instance to instance. The true order of objects (such as parts of a name, or
2595 lines in an address, or parts of a phone number) may need an explicit method to
2596 define their order.

2597 In this definition, the term "presented" may mean presentation to the user,
2598 reports, or transfer to other data systems. It is meaningful only when the order of
2599 appearance of items within a sequence is expressed. Such an order is only the
2600 default for the content within an instance. It may be overruled by any meaningful
2601 sorting or other processing.

2602 **[Rule 7-7]**

2603 Within a NIEM-conformant schema or instance, the attribute
2604 `structures:sequenceID` SHALL NOT be interpreted as meaningful beyond
2605 an indicator of sequential order of an object relative to its siblings.

2606 **Rationale**

2607 Siblings of a data item are items that have the same parent. Note that, using the
2608 reference and relationships mechanisms, data objects may have multiple
2609 parents. The `sequenceID` is truly metadata, helping to express the structure of
2610 the data, rather than its content.

2611 Note that reference elements have the same semantics as concrete data elements, and
2612 so follow the same rules for sequential order. By using reference elements, an entity may
2613 have one order within one structure, and another order within another structure.

2614 Within NIEM-conformant instances, the order of objects is found be given by sorting the
2615 objects by numerical value of their respective attribute `structures:sequenceID`, from
2616 smallest to highest. The relative order of objects with equal values for
2617 `structures:sequenceID` is their order within the XML instance. Objects with no value
2618 for `structures:sequenceID` occur after all objects that have values for
2619 `structures:sequenceID`, in their relative order within the XML instance.

2620 The use of instance-based sequencing, including the use of `structures:sequenceID`,
2621 is preferred over efforts to sequence data definitions. For example, the use of "address
2622 line 1", "address line 2", "address line 3", etc, is not recommended. Instead, a single
2623 "address line" would be preferred, with order expressed in the XML instance.

2624 **7.5. Instance Metadata**

2625 NIEM provides the metadata mechanism for giving information about object assertions.
2626 An object may have an attribute which refers to one or more metadata objects.

2627 **Example of metadata**

```
2628 <Person>
2629   <PersonName>
2630     <PersonGivenName structures:metadata="M1">John</PersonGivenName>
2631     <PersonGivenName structures:metadata="M2">Jack</PersonGivenName>
2632     <PersonSurName structures:metadata="M1 M2">Smith</PersonSurName>
2633   </PersonName>
2634   <PersonBirthDate>1945-12-01</PersonBirthDate>
2635 </Person>
2636 <Metadata structures:id="M1"><SourceText>Adam Barber</SourceText></Metadata>
2637 <Metadata structures:id="M2"><SourceText>Charles
2638 Daniels</SourceText></Metadata>
```

2639 This example shows a person. In this example, Adam Barber says the person is John
2640 Smith. Charles Daniels says his name is Jack Smith. A source for the person's birth date
2641 is not given.

2642 This shows several characteristics of metadata:

- 2643 1. Metadata objects may appear outside the data they describe
- 2644 2. Metadata objects may be reused
- 2645 3. Data may refer to more than one metadata object

2646 **[Rule 7-8]**

2647 Within a NIEM-conformant element instance, when an object O links to a
2648 metadata object via an attribute `structures:metadata`, the information in the
2649 metadata object SHALL be applied to the object O.

2650 **[Rule 7-9]**

2651 Within a NIEM-conformant element instance, when an object O1 contains an
2652 element E, with content object O2, and O2 links to a metadata object via an
2653 attribute `structures:linkMetadata`, the information in the metadata object
2654 SHALL be applied to the relationship E between O1 and O2.

2655 **Rationale**

2656 These two rules define the meaning of metadata:

- 2657 • `structures:metadata` applies metadata to an object.
- 2658 • `structures:linkMetadata` applies metadata to a relationship
2659 between two objects.

2660 **[Rule 7-10]**

2661 Within a NIEM-conformant element instance, each `IDREF` contained in the value
2662 of an attribute `structures:metadata` MUST refer to an attribute
2663 `structures:id` owned by an instance of a metadata type in the same
2664 information set.

2665 **[Rule 7-11]**

2666 Within a NIEM-conformant element instance, each `IDREF` contained in the value
2667 of an attribute `structures:linkMetadata` MUST refer to an attribute
2668 `structures:id` owned by an instance of a metadata type in the same
2669 information set.

2670 **Rationale**

2671 All `structures:metadata` and `structures:linkMetadata` attributes must
2672 refer to metadata objects.

2673 **[Rule 7-12]**

2674 Within a set of NIEM-conformant element instances within an information set,
2675 any metadata element instance referred to from an element instance of some
2676 type *T* MUST be applicable to an object type *T*.

2677 **Rationale**

2678 The applicability is determined by `structures:AppliesTo` application
2679 information of the metadata type definition. The instances must correspond to
2680 the types specified by the metadata type definition.

2681 8. Naming Rules

2682 This section outlines the rules used to create names for NIEM data components
2683 previously discussed in this document. Data component names must be understood
2684 easily both by humans and by machine processes. These rules improve name
2685 consistency by restricting characters, terms, and syntax that could otherwise allow too
2686 much variety and potential ambiguity. These rules also improve readability of names for
2687 humans, facilitate parsing of individual terms that compose names, and support various
2688 automated tasks associated with dictionary and controlled vocabulary maintenance.

2689 8.1. Extension of XSD Namespace Simple Types

2690 [Rule 8-0.9]

2691 Within a NIEM-conformant schema, a complex type that is a direct extension of
2692 an XML Schema namespace simple type MAY use the same local name as the
2693 simple type, if and only if the extension adds no content other than the attribute
2694 group `structures:SimpleObjectAttributeGroup`.

2695 Rationale

2696 It is useful to build complex type bases for further extension. The NIEM
2697 distribution proxy schema `xsd.xsd` provides complex type bases for some of the
2698 simple types in the XML Schema namespace. However, the complex types in
2699 this proxy schema reuse the local names of the simple types they extend, even
2700 though the simple type names may not be NIEM-conformant. Requiring name
2701 changes for those NIEM-provided complex type bases would work against user
2702 understanding, for those already familiar with the names of the XML Schema
2703 namespace simple types being extended.

2704 8.2. Usage of English

2705 [Rule 8-1]

2706 The name of any XML Schema component defined by NIEM-conformant
2707 schemas SHALL be composed of words from the English language, using the
2708 prevalent U.S. spelling, as provided by [OED].

2709 Rationale

2710 The English language has many spelling variations for the same word. For
2711 example, American English “program” has a corresponding British spelling
2712 “programme.” This variation has the potential to cause interoperability problems
2713 when exchanging XML components because of the different names used by the
2714 same elements. Providing a dictionary standard for spelling will mitigate this
2715 potential interoperability issue.

2716 8.3. Characters in Names

2717 [Rule 8-2]

2718 The name of any XML Schema component defined by a NIEM-conformant
2719 schema SHALL contain only the following characters:

- 2720 • upper-case letters ('A'-'Z'),
- 2721 • lower-case letters ('a'-'z'),
- 2722 • digits ('0'-'9'), and
- 2723 • hyphen ('-').

2724 Other characters, such as the underscore ('_') character and the period ('.')
2725 character SHALL NOT appear in component names in NIEM-conformant
2726 schemas.

2727 **[Rule 8-3]**

2728 The hyphen character ('-') MAY appear in component names only when used as
2729 a separator between parts of a single word, phrase, or value, that would
2730 otherwise be incomprehensible without the use of a separator.

2731 **Rationale**

2732 Names of standards and specifications, in particular, tend to consist of series of
2733 discrete numbers. Such names require some explicit separator, to keep the
2734 values from running together. The separator used within NIEM is the hyphen.

2735 Names of NIEM components follow the rules of XML Schema, by **[Rule 4-3]**. NIEM
2736 components also must follow the rules specified for each type of XML Schema
2737 component.

2738 **8.4. Character Case**

2739 **[Rule 8-4]**

2740 Within a NIEM-conformant schema, any attribute declaration SHALL have a
2741 name that begins with a lower-case letter ('a'-'z').

2742 **[Rule 8-5]**

2743 Within a NIEM-conformant schema, any XML Schema component other than an
2744 attribute declaration SHALL have a name that begins with an upper-case letter
2745 ('A'-'Z').

2746 Camel case is the practice of writing compound words or phrases in which the words are
2747 joined without spaces and are capitalized within the compound words.²

2748 **[Rule 8-6]**

2749 The name of any XML Schema component defined by a NIEM-conformant
2750 schema SHALL use the camel case formatting convention.

2751 **Rationale**

2752 The foregoing rules establish *lowerCamelCase* for all NIEM components that are
2753 XML attributes, and *UpperCamelCase* for all NIEM components that are types,
2754 elements, or groups.

2755 **8.5. Use of Acronyms and Abbreviations**

2756 Acronyms and abbreviations have the ability to improve readability and comprehensibility
2757 of large, complex, or frequently-used terms. They also obscure meaning and impair
2758 understanding when their definition is not clear, or when they are used in judiciously.
2759 They should be used with great care. Acronyms and abbreviations that are used must be
2760 documented, and used consistently.

2761 **[Rule 8-7]**

2762 A NIEM-conformant schema MUST consistently use approved acronyms,
2763 abbreviations, and word truncations within defined names. The approved
2764 shortened forms are defined in Table 2: Abbreviations used in NIEM Core Names
2765 .

² Adapted from http://en.wikipedia.org/wiki/Camel_case

Table 2: Abbreviations used in NIEM Core Names

Abbreviation	Full Meaning
ANSI	American National Standards Institute
CMV	Commercial Motor Vehicle
DEA	Drug Enforcement Agency
DNA	Deoxyribonucleic Acid
FGI	Foreign Government Information
FIPS	Federal Information Processing Standard
IC	Intelligence Community
ID	Identifier
IP	Internet Protocol
ISO	International Standards Organization
LIS	NCIC code list for license state
LSTA	NCIC code list for state/country index
MCO	Manufacturer's Certificate of Origin
MGRS	Military Grid Reference System
MSRP	Manufacturer's Suggested Retail Price
NANP	North American Numbering Plan
NCIC	National Crime Information Center
NCTC	National Counter Terrorist Center
NIBRS	National Incident Based Reporting System
NLETS	The International Justice & Public Safety Information Sharing Network (formerly known as the National Law Enforcement Teletype System)
ORI	Organization Identifier (Orion)
RES	NCIC code list for registration state for boat registrations
RF	Radio Frequency
SIM	Subscriber Identity Module
SSN	Social Security Number
TYP	NCIC code list for gun type
TYPO	NCIC code list for ORI type
URI	Uniform Resource Identifier
US	United States
UTM	Universal Transverse Mercator
VIN	Vehicle Identification Number
VINA	Vehicle Identification Number Analysis

2767 Rationale

2768 Consistent, controlled, and documented abridged terms that are used frequently
 2769 and/or tend to be lengthy can support readability, clarity, and reduction of name
 2770 length.

2771 8.6. Word Forms**2772 [Rule 8-8]**

2773 A noun used as a term in a NIEM component **MUST** be used in singular form,
 2774 unless the concept itself is plural.

2775 **[Rule 8-9]**
2776 A verb used as a term in a NIEM component MUST be used in the present tense,
2777 unless the concept itself is past tense.

2778 **[Rule 8-10]**
2779 Articles, conjunctions and prepositions SHALL NOT be used in NIEM component
2780 names, except where they are required for clarity or by standard convention
2781 (e.g.; `PowerOfAttorneyCode`).

2782 **Rationale**
2783 Articles (e.g., a, an, the), conjunctions (e.g., and, or, but), and prepositions (e.g.,
2784 at, by, for, from, in, of, to) are all disallowed in NIEM component names. These
2785 rules constrain slight variations in word forms and types to improve consistency
2786 and reduce potentially ambiguous or confusing component names.

2787 **8.7. Name Generation**

2788 Elements in NIEM-conformant schemas are given names that follow a specific pattern.
2789 This pattern comes from **[ISO 11179 Part 5]**.

2790 **[Rule 8-11]**
2791 Except as specified elsewhere in this document, any element or attribute defined
2792 within a NIEM-conformant schema SHALL have a name which takes the form:

- 2793 • object class qualifier terms (0 or more)
- 2794 • an object class term (1)
- 2795 • property qualifier terms (0 or more)
- 2796 • a property term (1)
- 2797 • representation qualifier terms (0 or more)
- 2798 • a representation term (1).

2799 **Rationale**
2800 Consistent naming rules are helpful for users who wish to understand
2801 components with which they are unfamiliar, as well as for users to find
2802 components with known semantics. This rule establishes the basic structure for
2803 an element or attribute name, in line with the rules for names under **[ISO 11179**
2804 **Part 5]**.

2805 **8.8. Object Class Term**

2806 The NIEM adopts an object-oriented approach to representation of data. Object classes
2807 represent what **[ISO 11179 Part 5]** refers to as “things of interest in a universe of
2808 discourse that may be found in a model of that universe.” An object class or object term
2809 is a word that represents a class of real-world entities or concepts. An object class term
2810 describes the applicable context for a NIEM component.

2811 **[Rule 8-12]**
2812 The object class term of a NIEM component SHALL consist of a term identifying
2813 a category of concrete concepts or entities.

2814 **Rationale**
2815 The object class term indicates the object category which this data component
2816 describes or represents. This term provides valuable context and narrows the
2817 scope of the component to an actual class of things or concepts.

2818 **Example**
2819 Concept term: Activity
2820 Entity term: Vehicle

2821 **8.9. Property Term**

2822 Objects or concepts are usually described in terms of their characteristic properties, data
2823 attributes, or constituent subparts. Most objects can be described by several
2824 characteristics. Therefore, a property term in the name of a data component represents a
2825 characteristic or subpart of an object class, and generally describes the essence of that
2826 data component.

2827 **[Rule 8-13]**

2828 A property term SHALL describe or represent a characteristic or subpart of an
2829 entity or concept.

2830 **Rationale**

2831 The property term describes the central meaning of the data component.

2832 **8.10. Qualifier Terms**

2833 Qualifier terms modify object, property, representation, or other qualifier terms in order to
2834 increase semantic precision and reduce ambiguity. Qualifier terms may precede or
2835 succeed the terms they modify. The goal for the placement of qualifier terms is to
2836 generally follow the rules of ordinary English while maintaining clarity.

2837 **[Rule 8-14]**

2838 Multiple qualifier terms MAY be used within a component name as necessary to
2839 ensure clarity and uniqueness within its namespace and usage context.

2840 **[Rule 8-15]**

2841 The number of qualifier terms SHOULD be limited to the absolute minimum
2842 required to make the component name unique and understandable.

2843 **[Rule 8-16]**

2844 The order of qualifiers SHALL NOT be used to differentiate names.

2845 **Rationale**

2846 Very large vocabularies may have many similar and closely related properties
2847 and concepts. The use of object, property, and representation terms alone is
2848 often not sufficient to construct meaningful names that can uniquely distinguish
2849 such components. Qualifier terms provide additional context to resolve these
2850 subtleties. However, swapping the order of qualifiers rarely (if ever) changes
2851 meaning; qualifier ordering is no substitute for meaningful terms.

2852 **8.11. Representation Term**

2853 The representation term for a component name serves several purposes in NIEM:

- 2854 1. It can indicate the style of component. For example, types are clearly labeled
2855 with the representation term `Type`.
- 2856 2. It helps prevent name conflicts and confusion. For example, elements and types
2857 may not be given the same name.

2858 3. It indicates the nature of the value carried by element. Labeling elements and
 2859 attributes with a notional indicator of the content eases discovery and
 2860 comprehension.

2861 **[Rule 8-17]**

2862 If any word in the representation term is redundant with any word in the property
 2863 term, one occurrence SHOULD be deleted.

2864 The valid value set of a data element or value domain is described by the representation
 2865 term. NIEM uses a standard set of representation terms in the representation portion of a
 2866 NIEM-conformant component name. Table 3: Representation Terms lists the primary
 2867 representation terms and a definition for the concept associated with the use of that term.
 2868 The table also lists secondary representation terms that may represent more specific
 2869 uses of the concept associated with the primary representation term.

2870 **Table 3: Representation Terms**

Primary Representation Term	Secondary Representation Term	Definition
Amount	-	A number of monetary units specified in a currency where the unit of currency is explicit or implied.
BinaryObject	-	A set of finite-length sequences of binary octets.
	Graphic	A diagram, graph, mathematical curves, or similar representation
	Picture	A visual representation of a person, object, or scene
	Sound	A representation for audio
	Video	A motion picture representation; may include audio encoded within
Code		A character string (letters, figures or symbols) that for brevity, language independence, or precision, represents a definitive value of an attribute.
DateTime		A particular point in the progression of time together with relevant supplementary information.
	Date	A particular day, month, and year in the Gregorian calendar.
	Time	A particular point in the progression of time within an unspecified 24 hour day.
ID		A character string to identify and distinguish uniquely, one instance of an object in an identification scheme from all other objects in the same scheme together with relevant supplementary information.

	URI	A string of characters used to identify (or name) a resource. The main purpose of this identifier is to enable interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols. A URI is either a Uniform Resource Locator (URL) or a Uniform Resource Name (URN). The specific syntax for each is defined by [RFC3986] .
Indicator		A list of two mutually exclusive Boolean values that express the only possible states of a property.
Measure		A numeric value determined by measuring an object along with the specified unit of measure.
Numeric		Numeric information that is assigned or is determined by calculation, counting, or sequencing. It does not require a unit of quantity or unit of measure.
	Value	A result of a calculation
	Rate	A representation of a ratio where the two units are not included.
	Percent	A representation of a ratio in which the two units are the same.
Quantity		A counted number of non-monetary units possibly including fractions.
Text	-	A character string (i.e. a finite sequence of characters) generally in the form of words of a language.
	Name	A word or phrase that constitutes the distinctive designation of a person, place, thing or concept.

2871 **[Rule 8-18]**

2872 Within a NIEM-conformant schema, the name of an element declaration that is of
2873 simple content **MUST** use a representation term found in Table 3: Representation
2874 Terms.

2875 **[Rule 8-19]**

2876 Within a NIEM-conformant schema, the name of an element declaration that is of
2877 complex content, and which corresponds to a concept listed in Table 3:
2878 Representation Terms, **MUST** use a representation term from that table.

2879 **[Rule 8-20]**

2880 Within a NIEM-conformant schema, the name of an element declaration which is
2881 of complex content and which does not correspond to a concept listed in Table 3:
2882 Representation Terms, MUST NOT use a representation term from that table.

2883 **[Rule 8-21]**

2884 Within a NIEM-conformant schema, the name of an attribute declaration MUST
2885 use a representation term from Table 3: Representation Terms.

2886 **Rationale**

2887 An element which represents a value listed in the table should have a
2888 representation term. It should do so even if its type is complex with multiple parts.
2889 For example, a type with multiple fields may represent a sound binary, or a date,
2890 or a name.

2891 **8.12. NIEM Type Names**

2892 This section contains naming rules specific to various kinds of NIEM types.

2893 **8.12.1. All Type Components**

2894 **[Rule 8-22]**

2895 Within a NIEM-conformant schema, the name of any type definition MUST use
2896 the representation term `Type`.

2897 **Rationale**

2898 Using the representation term `Type` immediately identifies XML types in a NIEM-
2899 conformant schema and prevents naming collisions with corresponding XML
2900 elements and attributes.

2901 **8.12.2. Simple Type Components**

2902 **[Rule 8-23]**

2903 Within a NIEM-conformant schema, the name of any simple type definition
2904 SHALL use the representation term qualifier `Simple`. This qualifier SHALL
2905 appear after any other representation term qualifiers.

2906 **Rationale**

2907 Specific uses of type definitions have similar syntax, but very different effects on
2908 data definitions. Schemas that clearly identify complex and simple type
2909 definitions are easier to understand without tool support. This rule ensures that
2910 names of simple types end in `SimpleType`.

2911 **8.12.3. Code Type Components**

2912 **[Definition: code type]**

2913 A **code type** is a simple type schema component definition which contains
2914 multiple `xsd:enumeration` facets.

2915 These types represent lists of values, each of which has a known meaning beyond the
2916 text representation. These values may be meaningful text or may be a string of
2917 alphanumeric identifiers which represent abbreviations for literals.

2918 **[Rule 8-24]**

2919 Within a NIEM-conformant schema, the name of any code type SHALL use the
2920 representation term qualifier `Code`.

2921 **Rationale**

2922 Using the qualifier `Code` (i.e. `CodeType`, `CodeSimpleType`) immediately
2923 identifies a type as representing a fixed list of codes. These types may be
2924 handled in specific ways, as lists of codes are expected to have their own
2925 lifecycles, including versions and periodic updates. Codes may also have
2926 responsible authorities behind them, who provide concrete semantic bindings for
2927 the code values.

2928 **[Rule 8-25]**

2929 Within a NIEM-conformant schema, any type definition which has a base type
2930 definition of a code type or which is transitively based on a code type SHALL
2931 have a name which uses the representation term qualifier `Code`.

2932 **Rationale**

2933 This expands the use of the representation term qualifier `Code` to any type based
2934 on a code list.

2935 **8.12.4. Association Type Components**

2936 **[Rule 8-26]**

2937 Within a NIEM-conformant schema, any association type SHALL have a name
2938 that uses the representation term qualifier `Association`. Types other than
2939 association types SHALL NOT use the representation term qualifier
2940 `Association`.

2941 **Rationale**

2942 Using the qualifier `Association` immediately identifies a type as representing
2943 an association.

2944 **8.12.5. Augmentation Type Components**

2945 **[Rule 8-27]**

2946 Within a NIEM-conformant schema, any augmentation type SHALL have a name
2947 that uses the representation term qualifier `Augmentation`. Types other than
2948 augmentation types SHALL NOT use the representation term qualifier
2949 `Augmentation`.

2950 **Rationale**

2951 Using the qualifier `Augmentation` immediately identifies a type as representing
2952 an augmentation.

2953 **8.12.6. Metadata Type Components**

2954 **[Rule 8-28]**

2955 Within a NIEM-conformant schema, any metadata type SHALL have a name that
2956 uses the representation term qualifier `Metadata`. Types other than metadata
2957 types SHALL NOT use the representation term qualifier `Metadata`.

2958 **Rationale**

2959 Using the qualifier `Metadata` immediately identifies a type as representing
2960 metadata.

2961 **8.13. NIEM Property Names**

2962 This section contains naming rules specific to different kinds of NIEM properties.

2963 **8.13.1. Attribute Group Names**

2964 **[Rule 8-29]**

2965 Within a NIEM-conformant schema, the name of any attribute group definition
2966 schema component SHALL use the representation term `AttributeGroup`.

2967 **Rationale**

2968 This clearly identifies attribute groups, and partitions their names from the names
2969 of other types of schema components.

2970 **8.13.2. Reference Names**

2971 **[Rule 8-30]**

2972 Within a NIEM-conformant schema, the name of any reference element SHALL
2973 use the representation term suffix `Reference`.

2974 **Rationale**

2975 Reference elements are identical in semantics to elements that are not by-
2976 reference. However, they refer to their values by a reference attribute, instead of
2977 carrying it as content of the XML element. The use of a suffix helps indicate that
2978 the elements refer to, instead of contain, their values, yet allows the basic
2979 semantics (e.g. property, representation term) to persist.

2980 Note that the use of the representation term suffix is one of the situations in
2981 which there is a slight divergence from the general rule for name generation as
2982 discussed in **[Rule 8-11]**.

2983 **8.13.3. Association Names**

2984 **[Rule 8-31]**

2985 Within a NIEM-conformant schema, the name of an association element SHALL
2986 use the representation term qualifier `Association`.

2987 **Rationale**

2988 Using the qualifier `Association` immediately identifies an element as
2989 representing an association.

2990 **8.13.4. Augmentation Names**

2991 **[Rule 8-32]**

2992 Within a NIEM-conformant schema, the name of an augmentation element
2993 SHALL use the representation term `Augmentation`.

2994 **Rationale**

2995 Using the qualifier `Augmentation` immediately identifies an element as
2996 representing an augmentation.

2997 **8.13.5. Metadata Names**

2998 **[Rule 8-33]**

2999 Within a NIEM-conformant schema, the name of a metadata element SHALL use
3000 the representation term `Metadata`.

3001 **Rationale**

3002 Using the qualifier `Metadata` immediately identifies an element as representing
3003 metadata.

3004 **8.13.6. Role Names**

3005 **[Rule 8-34]**

3006 Within a NIEM-conformant schema, the name of a role SHALL use the property
3007 term `RoleOf`.

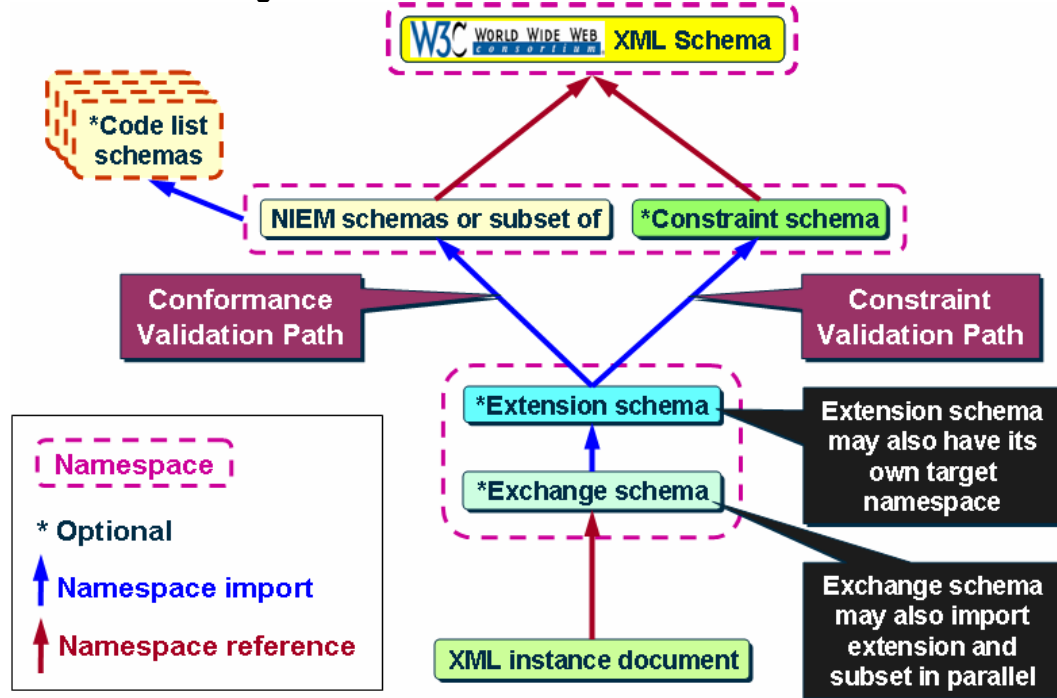
3008 **Rationale**

3009 Using the property term `RoleOf` immediately identifies an element as
3010 representing a role.

Appendix A. NIEM Overview

The NIEM is a reference model of unconstrained components rendered in XML Schema. Associated with the NIEM schemas is an XML reference architecture that organizes and guides the employment of the various kinds of schemas that compose a NIEM information exchange. The XML reference architecture describes the relationships between XML schemas for NIEM Information Exchange Package Documentation (IEPD).

Figure 1: The NIEM XML Reference Architecture



A NIEM IEPD is a set of artifacts that describe an Information Exchange Package (IEP), a standard message structure as defined by the Federal Enterprise Architecture Consolidated Reference Model Document [CRM]. The NIEM IEPD Specification [IEPD] contains a more detailed explanation of IEPDs and their contents.

The following kinds of XML schemas are associated with the NIEM reference architecture

- NIEM reference schemas: Schemas containing content created or approved by the NIEM steering committees are periodically released in schema distributions. The structure and content of such distributions are not specified in this document. This document specifies rules that apply to the NIEM-conformant schemas that are released as part of such distributions.
- NIEM support schemas: NIEM includes two special schemas, the `appinfo` and the `structures` schemas, for annotating and structuring NIEM-conformant schemas.
- Extension Schema: a NIEM-conformant schema which adds domain- or application-specific content to the base NIEM model.
- Exchange Schema: a NIEM-conformant schema which specifies a document in a particular exchange.
- Subset Schema: a profile of a NIEM-conformant schema, derived from a reference schema, but which specifies instances that only require a portion of the reference schema.

3039 • Constraint Schema: a schema which adds additional constraints to NIEM-
3040 conformant instances, but which is assumed to validate in concert with existing
3041 NIEM-conformant or subset schemas. A constraint schema need not validate
3042 constraints that are applied by other schemas.

3043 The only mandatory schemas for validation are the NIEM reference schemas or correct
3044 subsets. The NIEM schemas may import additional schemas, such as code table
3045 schemas, as needed. The optional exchange schema imports, re-uses, and organizes
3046 the components from the NIEM for the particular exchange. An optional extension
3047 schema may be used to add extended types and properties for components not
3048 contained in the NIEM, but which are needed for the exchange.

3049 Note that while only the reference schemas, or subsets thereof, are required for
3050 validation of a NIEM-conformant instance. The IEPD specification requires that an IEPD
3051 include an exchange schema along with the reference schemas (or subsets) to be
3052 considered a complete IEPD.

3053 The exchange and extension schemas can be combined into a single schema and
3054 namespace, or can be broken out into separate schemas and corresponding
3055 namespaces. The user may decide the best way to organize components. If the
3056 extension components will be reused elsewhere, it may be more efficient to maintain
3057 them in a separate namespace, rather than including them in a document namespace.

3058 The NIEM reference schemas are over-inclusive and under-constrained. The reason for
3059 this approach is that pre-determining all user needs and constraints is rarely possible.
3060 The only way to reach consensus on components is to include all obvious requirements
3061 and maintain relatively relaxed constraints.

3062 To ensure interoperability, specific component requirements and constraints are
3063 determined on a per-exchange basis (in IEPDs). By creating a subset of NIEM Core,
3064 reference and code table schemas, the user can limit the components to only those he or
3065 she needs. In the future, a business component layer between IEPDs and NIEM will
3066 allow domains to apply consistent requirements and constraints for their exchanges.

3067 The basic principle for a subset is that an instance that validates against a correct subset
3068 schema will always validate against the full reference NIEM schema set. The user may
3069 also adjust cardinality constraints, as desired, within the subset schemas.

3070 Additional constraints may be handled in a constraint schema. A constraint schema is
3071 derived from a subset schema. However, it may contain other constraints (for example,
3072 additional types for specific constraints). The constraint schema provides an alternative
3073 *constraint validation* path that allows the user to reduce the possible set of allowable XML
3074 instances, independent of the NIEM schema or subset *conformance validation* path. This
3075 is done through multi-pass validation. A correctly constructed XML instance will validate
3076 through both the conformance and the constraint path.

3077 **Appendix B. NIEM Design Principles**

3078 This appendix summarizes all the underlying NIEM design principles discussed in
3079 Section 3, Guiding Principles.

3080 **[Principle 1]**

3081 This specification should specify what is necessary for interoperability, and no more.

3082 **[Principle 2]**

3083 This specification should focus on providing rules for specifying schemas.

3084 **[Principle 3]**

3085 This specification should feature rules which are as specific, precise, and concise as
3086 possible.

3087 **[Principle 4]**

3088 The content of a NIEM-conformant data instance should not be modified by processing
3089 against XML schemas.

3090 **[Principle 5]**

3091 NIEM should depend on XML Schema validating parsers for validation of XML content.

3092 **[Principle 6]**

3093 The primary purpose of XML Schema validation is to restrict processed data to that data
3094 that conforms to agreed-upon rules. This restriction is achieved by marking as invalid
3095 that data that does not conform to the rules defined by the schema.

3096 **[Principle 7]**

3097 Constraints on XML instances MAY be validated by multiple schema validation passes,
3098 using multiple schemas for a single namespace.

3099 **[Principle 8]**

3100 Each NIEM-conformant namespace will be defined by exactly one reference schema.

3101 **[Principle 9]**

3102 NIEM-conformant schemas do not specify data that uses mixed content.

3103 **[Principle 10]**

3104 Using named global components in schemas maximizes the capacity for reuse.

3105 **[Principle 11]**

3106 Wildcards in standard schemas should be avoided.

3107 **[Principle 12]**

3108 Schema locations specified within NIEM-conformant reference schemas are hints and
3109 provide default values to processing applications.

3110 **[Principle 13]**

3111 NIEM-conformant instances and schemas should reuse components from NIEM
3112 distribution schemas when possible.

3113 **[Principle 14]**

3114 A namespace is a required part of the name of a component. A component's local name
3115 is considered independent of, and unassociated with, names from other namespaces.

3116 **[Principle 15]**
3117 NIEM is intended for extension and augmentation by users and developers outside the
3118 standardization process.

3119 **[Principle 16]**
3120 XML data is primarily intended for automatic processing, not for literal presentation to
3121 people.

3122 **[Principle 17]**
3123 NIEM should not depend on specific software packages, frameworks, or systems for
3124 interpretation of XML instances.

3125 **[Principle 18]**
3126 NIEM should be implemented with a variety of commercial off-the-shelf and free software
3127 products.

3128 **[Principle 19]**
3129 A data component definition should be drafted before the associated data element name
3130 is composed.

3131 **[Principle 20]**
3132 Components in NIEM should be given names which are consistent with names of other
3133 NIEM components. Such names should be based on simple rules.
3134

3135 **Appendix C. NIEM Rules**

3136 This listing of rules is informative only. For reference purposes, it summarizes all the
3137 rules found in this document.

3138 **[Rule 4-1]**

3139 A NIEM-conformant schema MUST conform to XML as specified by **[XML]**

3140 **[Rule 4-2]**

3141 A NIEM-conformant schema MUST conform to the specification for namespaces
3142 in XML, as defined by **[XMLNamespaces]** and **[XMLNamespacesErrata]**.

3143 **[Rule 4-3]**

3144 A NIEM-conformant schema MUST conform to the W3C XML Schema
3145 Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Datatypes,
3146 as specified by **[XMLSchemaStructures]** and **[XMLSchemaDatatypes]**.

3147 **[Rule 4-4]**

3148 Within a NIEM-conformant schema, the text definition provided for each
3149 documented component SHALL follow the requirements and recommendations for data
3150 definitions given by **[ISO 11179 Part 4]**.

3151 **[Rule 4-5]**

3152 In general, a NIEM component name SHALL be formed by applying the
3153 informative guidelines and examples detailed in Annex A of **[ISO 11179 Part 5]**, with
3154 exceptions as specified in this document, most notably those specified in Section 8,
3155 Naming Rules.

3156 **[Rule 5-1]**

3157 Within a NIEM-conformant schema, an element `xsd:complexType` SHALL
3158 NOT own the attribute `mixed` with the value `true`.

3159 **[Rule 5-2]**

3160 Within a NIEM-conformant schema, an element declaration which is of complex
3161 content SHALL NOT own the attribute `mixed` with the value `true`.

3162 **[Rule 5-3]**

3163 A NIEM-conformant schema SHALL NOT contain a reference to the type
3164 definition `xsd:NOTATION`, or to a type derived from that type.

3165 **[Rule 5-4]**

3166 A NIEM-conformant schema SHALL NOT contain the element `xsd:notation`.

3167 **[Rule 5-5]**

3168 A NIEM-conformant schema SHALL NOT contain the element `xsd:include`.

3169 **[Rule 5-6]**

3170 A NIEM-conformant schema SHALL NOT contain the element `xsd:redefine`.

3171 **[Rule 5-7]**

3172 A NIEM-conformant schema SHALL NOT reference the type `xsd:anyType`.

3173 **[Rule 5-8]**

3174 A NIEM-conformant schema SHALL NOT reference the type
3175 `xsd:anySimpleType`.

3176 **[Rule 5-9]**
3177 Within a NIEM-conformant schema, an element declaration with the attribute
3178 `name` and without the attribute `type` MUST carry the attribute `abstract` with the value
3179 `true`.

3180 **[Rule 5-10]**
3181 Within a NIEM-conformant schema, an attribute declaration with attribute `name`
3182 MUST carry the attribute `type`.

3183 **[Rule 5-11]**
3184 A NIEM-conformant schema SHALL NOT contain the element `xsd:any`.

3185 **[Rule 5-12]**
3186 A NIEM-conformant schema SHALL NOT contain the element
3187 `xsd:anyAttribute`.

3188 **[Rule 5-13]**
3189 Within a NIEM-conformant schema, any type definition MUST appear as an
3190 immediate child of the document element `xsd:schema`.

3191 **[Rule 5-14]**
3192 Within a NIEM-conformant schema, any element declaration carrying the
3193 attribute `name` MUST appear as an immediate child of the document element
3194 `xsd:schema`.

3195 **[Rule 5-15]**
3196 Within a NIEM-conformant schema, any attribute declaration owning the attribute
3197 `name` MUST appear as an immediate child of the document element `xsd:schema`.

3198 **[Rule 5-16]**
3199 A NIEM-conformant schema SHALL NOT contain any of the elements
3200 `xsd:unique`, `xsd:key`, `xsd:keyref`, `xsd:selector`, or `xsd:field`.

3201 **[Rule 5-17]**
3202 A NIEM-conformant schema SHALL NOT contain the element `xsd:all` or the
3203 element `xsd:choice`.

3204 **[Rule 5-18]**
3205 Within a NIEM-conformant schema, any immediate child of a model group
3206 `xsd:sequence` element MUST be one of `xsd:annotation`, or `xsd:element`.

3207 **[Rule 5-19]**
3208 A NIEM-conformant schema SHALL NOT contain the element `xsd:group`.

3209 **[Rule 5-20]**
3210 Within a NIEM-conformant schema, if the element `xsd:sequence` carries the
3211 attribute `minOccurs`, it MUST set the value for the attribute to 1.

3212 **[Rule 5-21]**
3213 Within a NIEM-conformant schema, if the element `xsd:sequence` carries the
3214 attribute `maxOccurs`, it MUST set the value of the attribute to 1.

3215 **[Rule 5-22]**
3216 Within a NIEM-conformant schema, if an element declaration carries the attribute
3217 `block`, it MUST set the value for the attribute to the empty string.

3218 **[Rule 5-23]**
3219 Within a NIEM-conformant schema, if a complex type definition carries the
3220 attribute `block`, it MUST set the value for the attribute to the empty string.

3221 **[Rule 5-24]**
3222 Within a NIEM-conformant schema, if the document element `xsd:schema`
3223 carries the attribute `blockDefault`, it MUST set the value for the attribute to the empty
3224 string.

3225 **[Rule 5-25]**
3226 Within a NIEM-conformant schema, if a simple type definition carries the attribute
3227 `final`, it MUST set the value for the attribute to the empty string.

3228 **[Rule 5-26]**
3229 Within a NIEM-conformant schema, if a complex type definition carries the
3230 attribute `final`, it MUST set the value for the attribute to the empty string.

3231 **[Rule 5-27]**
3232 Within a NIEM-conformant schema, if an element declaration carries the attribute
3233 `final`, it MUST set the value for the attribute to the empty string.

3234 **[Rule 5-28]**
3235 Within a NIEM-conformant schema, if the document element `xsd:schema`
3236 carries the attribute `finalDefault`, it MUST set the value for that attribute to the empty
3237 string.

3238 **[Rule 5-29]**
3239 Within a NIEM-conformant schema, any element `xsd:element` SHALL NOT
3240 carry the attribute `default`.

3241 **[Rule 5-30]**
3242 Within a NIEM-conformant schema, any element `xsd:attribute` SHALL NOT
3243 carry the attribute `default`.

3244 **[Rule 5-31]**
3245 A NIEM-conformant schema SHALL NOT contain the element `xsd:list`.

3246 **[Rule 5-32]**
3247 A NIEM-conformant schema SHALL NOT contain the element `xsd:union`.

3248 **[Rule 5-33]**
3249 Within a NIEM-conformant schema, the document element `xsd:schema` MUST
3250 carry the attribute `targetNamespace`.

3251 **[Rule 5-34]**
3252 The value of the required attribute `targetNamespace` on the document element
3253 `xsd:schema` MUST match the production `<absolute-URI>` as defined by **[RFC3986]**.

3254 **[Rule 5-35]**

3255 Within a NIEM-conformant schema, the document element `xsd:schema` MUST

3256 carry the attribute `version`.

3257 **[Rule 5-36]**

3258 The value of the required attribute `version` on the document element

3259 `xsd:schema` MUST NOT be an empty string.

3260 **[Rule 5-37]**

3261 Within a NIEM-conformant schema, the element `xsd:import` MUST carry the

3262 attribute `namespace`.

3263 **[Rule 5-38]**

3264 The value of the required attribute `namespace` carried by the element

3265 `xsd:import` MUST match the production `<absolute-URI>` as defined by **[RFC3986]**.

3266 **[Rule 5-39]**

3267 Within a NIEM-conformant schema, the element `xsd:import` MUST carry the

3268 attribute `schemaLocation`.

3269 **[Rule 5-41]**

3270 Within a NIEM-conformant schema, the value of the required attribute

3271 `schemaLocation` carried by the element `xsd:import` MUST match either the

3272 production `<absolute-URI>`, or the definition of "*relative-path reference*", as defined

3273 by **[RFC3986]**.

3274 **[Rule 5-42]**

3275 Within a NIEM-conformant schema, the value of the required attribute

3276 `schemaLocation` carried by the element `xsd:import` MUST be resolvable to a XML

3277 schema document file that is valid according to **[XMLSchemaStructures]** and

3278 **[XMLSchemaDatatypes]**.

3279 **[Rule 5-43]**

3280 Within a NIEM-conformant schema, when a namespace other than the XML

3281 namespace or the XML Schema namespace is used, it MUST be imported into the

3282 schema using the `xsd:import` element.

3283 **[Rule 5-44]**

3284 Within a NIEM-conformant schema, when a namespace other than the XML

3285 namespace or the XML Schema namespace is used, its content MUST be valid with

3286 respect to the schema imported for that namespace.

3287 **[Rule 5-45]**

3288 Within a NIEM-conformant schema, an element SHALL have at most one

3289 instance of an element `xsd:annotation` as an immediate child.

3290 **[Rule 5-46]**

3291 Within a NIEM-conformant schema, the content of an `xsd:documentation`

3292 element MUST be character information items as specified by **[XMLInfoSet]**.

3293 **[Rule 5-47]**

3294 Within a NIEM-conformant schema, the element `xsd:annotation` MUST have

3295 at most one instance of the element `xsd:documentation` as an immediate child.

3296 **[Rule 5-48]**
3297 XML comments SHALL not be used for persistent information about constructs
3298 within XML Schemas.

3299 **[Rule 5-49]**
3300 Within a NIEM-conformant schema, any immediate child of an `xsd:appinfo`
3301 element SHALL be an element information item, or a comment information item.

3302 **[Rule 5-50]**
3303 Within a NIEM-conformant schema, any element that is an immediate child of an
3304 `xsd:appinfo` element SHALL be in a namespace.

3305 **[Rule 5-50.1]**
3306 Within a NIEM-conformant schema, an element in the XML Schema namespace
3307 MUST NOT occur as a descendant of any element `xsd:appinfo`.

3308 **[Rule 5-51]**
3309 Within NIEM-conformant schemas, the element `xsd:simpleType` MUST have
3310 the element `xsd:restriction` as an immediate child.

3311 **[Rule 5-52]**
3312 Within a NIEM-conformant schema, the element `xsd:complexType` MUST
3313 have as an immediate child either the element `xsd:complexContent` or the element
3314 `xsd:simpleContent`.

3315 **[Rule 5-53]**
3316 Within a NIEM-conformant schema, the element `xsd:simpleContent` MUST
3317 have as an immediate child the element `xsd:extension`.

3318 **[Rule 5-54]**
3319 Within a NIEM-conformant schema, given an element `xsd:simpleContent`
3320 with a child `xsd:extension` owning an attribute `base`, if the attribute `base` has a value
3321 that resolves to the name of a simple type, then the element `xsd:extension` MUST
3322 have an immediate child element `xsd:attributeGroup`.

3323 **[Rule 5-55]**
3324 Within a NIEM-conformant schema, the element `xsd:complexContent` MUST
3325 have as an immediate child the element `xsd:extension`.

3326 **[Rule 5-56]**
3327 Within a NIEM-conformant schema, given an element `xsd:complexContent`
3328 with a child `xsd:extension` owning an attribute `base`, the attribute `base`
3329 MUST have a value that resolves to the name of one of

- 3330 1. the type `structures:ComplexObjectType`, or
- 3331 2. the type `structures:MetadataType`, or
- 3332 3. the type `structures:AugmentationType`, or
- 3333 4. a NIEM-conformant complex type.

3334 **[Rule 5-57]**
3335 Within a NIEM-conformant schema, any occurrence of the element
3336 `xsd:attributeGroup` MUST own an attribute `ref`.

3337 **[Rule 5-58]**

3338 Within a NIEM-conformant schema, the attribute `ref` owned by any element
3339 `xsd:attributeGroup` MUST have a value of a qualified name (possibly using the
3340 default namespace) that SHALL resolve to the namespace for the NIEM structures
3341 namespace and the local name `SimpleObjectAttributeGroup`.

3342 **[Rule 6-1]**

3343 Within a NIEM-conformant schema, the document element `xsd:schema` MUST
3344 have application information `appinfo:ConformantIndicator`, with text content
3345 "true".

3346 **[Rule 6-2]**

3347 Two XML schemas SHALL have the same value for attribute `targetNamespace`
3348 carried by the element `xsd:schema` if and only if they represent the same set of
3349 components.

3350 **[Rule 6-3]**

3351 Two XML Schemas SHALL have the same value for attribute
3352 `targetNamespace` carried by the element `xsd:schema`, and different values for
3353 attribute `version` carried by the element `xsd:schema` if and only if they are different
3354 views of the same set of components.

3355 **[Rule 6-4]**

3356 Within a NIEM-conformant schema, any type definition MUST be a documented
3357 component.

3358 **[Rule 6-5]**

3359 Within a NIEM-conformant schema, any element declaration MUST be a
3360 documented component.

3361 **[Rule 6-6]**

3362 Within a NIEM-conformant schema, any attribute declaration MUST be a
3363 documented component.

3364 **[Rule 6-7]**

3365 Within a NIEM-conformant schema, the element `xsd:enumeration` MUST be a
3366 documented component.

3367 **[Rule 6-8]**

3368 Within a NIEM-conformant schema, the document element `xsd:schema` MUST
3369 be a documented component.

3370 **[Rule 6-9]**

3371 Words or synonyms for the words within a data element definition MAY be reused
3372 as terms in the corresponding component name, if those words do not dilute the
3373 semantics and understanding of, or impart ambiguity to, the entity or concept that the
3374 component represents.

3375 **[Rule 6-10]**

3376 An object class SHALL have one and only one associated semantic meaning (i.e.
3377 a single word sense.) as described in the definition of the component that represents that
3378 object class.

3379 **[Rule 6-11]**

3380 An object class SHALL NOT be redefined within the definitions of the
3381 components that represent properties or subparts of that entity or class.

3382 **[Rule 6-12]**

3383 A NIEM data definition SHALL NOT contain explicit representational or data
3384 typing information such as number characters, type of characters, etc., unless the very
3385 nature of the component can only be described by such information.

3386 **[Rule 6-13]**

3387 A component definition SHALL begin with a standard opening phrase that
3388 depends on the class of the component per Table 1: Standard Opening Phrases:

3389 **[Rule 6-14]**

3390 A NIEM-conformant schema SHALL import the `appinfo` namespace.

3391 **[Rule 6-15]**

3392 A component which is deprecated SHALL be indicated as such by the component
3393 having application information `appinfo:Deprecated`, with an attribute value with a
3394 value of `true`.

3395 **[Rule 6-16]**

3396 Within a NIEM-conformant schema, the element `appinfo:Base` MAY be used
3397 in one of the following ways:

- 3398 1. By a type definition, to indicate the base type, or `structures:Object`
3399 or `structures:Association`, or
3400 2. By an element declaration, to indicate the base element

3401 The element `appinfo:Base` SHALL NOT be used for any other purpose.

3402 **[Rule 6-17]**

3403 Within a NIEM-conformant schema, the element `appinfo:Base` SHALL
3404 indicate, by namespace and name, one of the following:

- 3405 1. a NIEM-conformant schema component, or
3406 2. `structures:Object`, or
3407 3. `structures:Association`.

3408 **[Rule 6-18]**

3409 Within a NIEM-conformation schema, an attribute `appinfo:namespace` owned
3410 by an element `appinfo:Base` SHALL have a value of either:

- 3411 1. a namespace which is the target namespace of a NIEM-conformant
3412 schema, or
3413 2. the `structures` namespace.

3414 **[Rule 6-19]**

3415 Within a NIEM-conformant schema, an element `appinfo:Base` which does not
3416 own an attribute `appinfo:namespace` SHALL refer to the target namespace of the
3417 schema in which it is used.

3418 **[Rule 6-20]**

3419 Within a NIEM-conformant schema, an element `appinfo:Base` SHALL own an
3420 attribute `appinfo:name`.

3421 **[Rule 6-21]**

3422 Within a NIEM-conformant schema, if an element `appinfo:Base` indicates a
3423 NIEM-conformant namespace, then the value of the attribute `appinfo:name` owned by
3424 the element `appinfo:Base` SHALL indicate a schema component in the indicated
3425 namespace.

3426 **[Rule 6-22]**

3427 Within a NIEM-conformant schema, if an element `appinfo:Base` indicates the
3428 structures namespace, then the value of the attribute `appinfo:name` owned by
3429 the element `appinfo:Base` SHALL have a value of one of:

- 3430 1. `structures:Object`, or
- 3431 2. `structures:Association`, or
- 3432 3. a schema component defined by the `structures` schema.

3433 **[Rule 6-23]**

3434 Within a NIEM-conformant schema, the element `appinfo:AppliesTo` MAY be
3435 used in any of the following ways:

- 3436 1. To indicate a base type to which an augmentation may be applied
- 3437 2. To indicate a base type to which a metadata type may be applied

3438 The element `appinfo:AppliesTo` SHALL NOT be used for any other purpose.

3439 **[Rule 6-24]**

3440 Within a NIEM-conformant schema, the element `appinfo:AppliesTo` SHALL
3441 indicate a schema component, by namespace and name.

3442 **[Rule 6-25]**

3443 Within a NIEM-conformation schema, an attribute `appinfo:namespace` owned
3444 by an element `appinfo:AppliesTo` SHALL indicate the namespace of the type to
3445 which `appinfo:AppliesTo` refers. The indicated namespace SHALL be NIEM-
3446 conformant.

3447 **[Rule 6-26]**

3448 The type to which the attribute `appinfo:appliesTo` refers MUST be the
3449 indicated type or MUST be transitively derived from the indicated type.

3450 **[Rule 6-27]**

3451 Within a NIEM-conformant schema, an element `appinfo:AppliesTo` which
3452 does not carry an attribute `appinfo:namespace` SHALL refer to the target namespace
3453 of the schema in which it is used.

3454 **[Rule 6-28]**

3455 Within a NIEM-conformant schema, an element `appinfo:AppliesTo` SHALL
3456 carry an attribute `appinfo:name`. The value of this attribute SHALL indicate the local
3457 name of a schema component within the namespace specified by the element.

3458 **[Rule 6-29]**

3459 Within a NIEM-conformant schema, the element `appinfo:ReferenceTarget`
3460 SHALL specify the type of a schema component which an instance of a reference
3461 element references. The element `appinfo:ReferenceTarget` SHALL NOT be used
3462 for any other purpose.

3463 **[Rule 6-30]**

3464 A reference element SHALL reference an instance of the indicated type, or an
3465 instance of a type derived from that type.

3466 **[Rule 6-30.1]**

3467 Within a NIEM-conformant schema, a reference element MUST have at most one
3468 instance of the element `appinfo:ReferenceTarget`.

3469 **[Rule 6-31]**

3470 Within a NIEM-conformant schema, the element `appinfo:ReferenceTarget`
3471 SHALL indicate a type definition schema component, by namespace and name.

3472 **[Rule 6-32]**

3473 Within a NIEM-conformation schema, an attribute `appinfo:namespace` carried
3474 by an element `appinfo:ReferenceTarget` SHALL indicate the namespace of the
3475 referenced schema component. The indicated namespace SHALL be NIEM-conformant.

3476 **[Rule 6-33]**

3477 Within a NIEM-conformant schema, an element `appinfo:ReferenceTarget`
3478 which does not carry an attribute `appinfo:namespace` SHALL refer to the target
3479 namespace of the schema in which it is used.

3480 **[Rule 6-34]**

3481 Within a NIEM-conformant schema, an element `appinfo:ReferenceTarget`
3482 SHALL carry an attribute `appinfo:name`. The value of this attribute SHALL indicate the
3483 local name of a type definition schema component within the namespace specified by the
3484 element.

3485 **[Rule 6-35]**

3486 Within a NIEM-conformant schema, a complex type definition SHALL be one of
3487 the following classes of types:

- 3488 1. An object type
- 3489 2. A role type
- 3490 3. An association type
- 3491 4. A metadata type
- 3492 5. An augmentation type
- 3493 6. An adapter type.

3494 **[Rule 6-36]**

3495 Within a NIEM-conformant schema, an element MUST NOT be introduced more
3496 than once into the direct content of a type definition. This applies to content acquired
3497 through extension of base types. This does not apply to a base element or derived
3498 element to one previously existing in the type definition.

3499 **[Rule 6-37]**

3500 Within a NIEM-conformant schema, an object type SHALL be a complex type
3501 definition that has one of the following forms:

- 3502 1. Has simple content, is based on a simple type, and contains the attribute
3503 group `structures:SimpleObjectAttributeGroup`, and has
3504 application information `appinfo:Base of structures:Object`, or
- 3505 2. Has complex content, and is based on complex type
3506 `structures:ComplexObjectType`, and has application information
3507 `appinfo:Base of structures:Object`, or
- 3508 3. Is a complex type that is derived from an object type, which is defined
3509 according to this rule.

3510 **[Rule 6-38]**

3511 Within a NIEM-conformant schema, any element with a name beginning with the
3512 string `RoleOf` SHALL represent a base type, of which the containing type represents a
3513 role.

3514 **[Rule 6-39]**

3515 Within a NIEM-conformant schema, an association type SHALL be a complex
3516 type definition that has one of the following forms:

- 3517 1. Has complex content, is based on the complex type
3518 `structures:ComplexObjectType`, and has application information
3519 `appinfo:Base of structures:Association`, or
- 3520 2. Is a complex type that is derived from an association type, which is
3521 defined according to this rule.

3522 **[Rule 6-40]**

3523 Within a NIEM-conformant schema, in an association type, any element which
3524 represents a participant in the relationship established by the association type SHALL be
3525 a reference element.

3526 **[Rule 6-41]**

3527 Within a NIEM-conformant schema, a metadata type SHALL contain elements
3528 appropriate for a specific class of data about data.

3529 **[Rule 6-42]**

3530 Within a NIEM-conformant schema, a metadata type and only a metadata type
3531 SHALL be derived directly from `structures:MetadataType`.

3532 **[Rule 6-43]**

3533 Within a NIEM-conformant schema, a metadata type MAY have application
3534 information `appinfo:AppliesTo`, indicating the NIEM-conformant object, association,
3535 or external adapter types to which the metadata applies.

3536 **[Rule 6-44]**

3537 Within a NIEM-conformant schema, a metadata type which does not have
3538 application information `appinfo:AppliesTo` MAY be applied to any object type,
3539 association type, or external adapter type.

3540 **[Rule 6-45]**

3541 An augmentation type:

- 3542 1. SHALL be transitively derived from `structures:AugmentationType`
3543 and
3544 2. SHALL contain elements which represent properties to be applied to a
3545 base type.

3546 **[Rule 6-46]**

3547 Within a NIEM-conformant schema, an augmentation element definition:

- 3548 1. SHALL have a type which is an augmentation type
3549 2. SHALL use the `substitutionGroup` attribute such that it is transitively
3550 substitutable for the element `structures:Augmentation`

3551 An element which is not an augmentation element SHALL NOT meet either of the
3552 above criteria.

3553 **[Rule 6-47]**

3554 Within a NIEM-conformant schema, an element definition for an augmentation
3555 element MAY contain one or more instances of the element `structures:AppliesTo`
3556 as application information, to specify types to which the augmentation element applies.

3557 **[Rule 6-48]**

3558 Within a NIEM-conformant schema, an element definition for an augmentation
3559 element which does not contain any instances of the element `structures:AppliesTo`
3560 MAY be applied to any object or association type.

3561 **[Rule 6-49]**

3562 Any type definition referenced by a component within a NIEM-conformant
3563 schema MUST be from one of the following:

- 3564 1. The schema being defined
3565 2. A namespace imported as NIEM-conformant
3566 3. The XML Schema namespace
3567 4. The `structures` namespace.

3568 **[Rule 6-50]**

3569 Any element declaration referenced by a component within a NIEM-conformant
3570 schema MUST be from one of the following:

- 3571 1. The schema being defined
3572 2. A namespace imported as NIEM-conformant
3573 3. The `structures` namespace
3574 4. An external namespace, in accordance with the rules for external
3575 schemas as specified by this specification.

3576 **[Rule 6-51]**

3577 Any attribute declaration referenced by a component within a NIEM-conformant
3578 schema MUST be from one of the following:

- 3579 1. The schema being defined
3580 2. A namespace imported as NIEM-conformant
3581 3. The `structures` namespace
3582 4. The XML namespace

3583 5. An external namespace, in accordance with the rules for external
3584 schemas as specified by this specification.

3585 **[Rule 6-52]**

3586 A NIEM-conformant schema MUST import the NIEM `structures` namespace.

3587 **[Rule 6-53]**

3588 NIEM-conformant schemas and instances MUST use content within the NIEM
3589 `structures` namespace as specified in this document and ONLY as specified by this
3590 document.

3591 **[Rule 6-54]**

3592 Within a NIEM-conformant schema, a complex type definition SHALL include the
3593 attribute `structures:sequenceID` if the order of an occurrence of the type, within its
3594 parent, relative to its siblings, is meaningful and pertinent, and if the content presented by
3595 all instances defined by the schema will not otherwise occur in the desired sequential
3596 order.

3597 **[Rule 6-55]**

3598 Within a NIEM-conformant schema, a reference element and only a reference
3599 element SHALL be defined to be of type `structures:ReferenceType`.

3600 **[Rule 6-56]**

3601 Within a NIEM-conformant schema, a complex type SHALL NOT be defined such
3602 that an instance of that type owns the attribute `structures:ref`.

3603 **[Rule 6-57]**

3604 Within a NIEM-conformant schema, any two elements of the form

3605 *NCName*

3606 and

3607 *NCNameReference*

3608 where the string value of *NCName* is the same in both forms, SHALL be defined
3609 to have identical semantics. The NIEM recognizes no difference in meaning between a
3610 reference element and an element that is not a reference element.

3611 **[Rule 6-58]**

3612 Within a NIEM-conformant schema, if both elements *NCName* and
3613 *NCNameReference* exist, then the `appinfo:ReferenceTarget` of any
3614 *NCNameReference* element MUST be the type of the element *NCName*.

3615 **[Rule 6-59]**

3616 Within a NIEM-conformant schema, an element `xsd:import` that imports a
3617 namespace defined by an external schema MUST have the application information
3618 `appinfo:ConformantIndicator`, with a value of `false`.

3619 **[Rule 6-60]**

3620 Within a NIEM-conformant schema, an element `xsd:import` that imports a
3621 namespace defined by an external schema MUST be a documented component.

3622 **[Rule 6-61]**

3623 Within a NIEM-conformant schema, an adapter type MUST have application
3624 information `appinfo:ExternalAdapterTypeIndicator` with a value of `true`. A
3625 type that is not an adapter type SHALL NOT contain that indicator.

3626 **[Rule 6-62]**

3627 Within a NIEM-conformant schema, an adapter type MUST be a immediate
3628 extension of type `structures:ComplexObjectType`.

3629 **[Rule 6-63]**

3630 Within a NIEM-conformant schema, an adapter type MUST be composed of only
3631 elements and attributes from an external standard.

3632 **[Rule 6-64]**

3633 Within a NIEM-conformant schema, an element reference used in an adapter
3634 type definition MUST be a documented component.

3635 **[Rule 6-65]**

3636 Within a NIEM-conformant schema, an attribute reference used in an adapter
3637 type definition MUST be a documented component.

3638 **[Rule 6-66]**

3639 Within a NIEM-conformant schema, an adapter type MUST NOT be extended or
3640 restricted.

3641 **[Rule 7-1]**

3642 A NIEM-conformant instance MUST validate to an authoritative NIEM-conformant
3643 schema set for namespaces contained in the instance, and for additional namespaces
3644 required for validation.

3645 **[Rule 7-2]**

3646 Within a NIEM-conformant instance, the meaning of an element with no content
3647 is that additional properties are not asserted. There SHALL NOT be additional meaning
3648 interpreted for an element with no content.

3649 **[Rule 7-3]**

3650 Within a NIEM-conformant element instance, there SHALL NOT be any
3651 difference in meaning between a property asserted via element containment and a
3652 property asserted by element reference, except as explicitly described by the semantics
3653 of the elements involved.

3654 **[Rule 7-4]**

3655 Any attribute `structures:ref` MUST have a value which occurs as the value
3656 of an attribute `structures:id` within the same information set.

3657 **[Rule 7-5]**

3658 Within a NIEM-conformant element instance, given that a reference element is
3659 restricted to a set S of target types T_i , $S = \{ T_1, T_2, \dots, T_n \}$, any attribute
3660 `structures:ref` MUST indicate the value of an attribute `structures:id` which is
3661 owned by an element of a type T such that T is, or is derived from, some type T_i in S.

3662 **[Rule 7-6]**

3663 The order of elements that are children of a NIEM-conformant element SHALL be
3664 presented as if their sequential order is as follows:

- 3665 1. First, elements owning an attribute `structures:sequenceID`, in the
3666 order that would be yielded with their sequence IDs sorted via XSLT's
3667 `sort` element, with a data type of `number` and an order of `ascending`.
- 3668 2. Following those elements, the remaining elements, in the order in which
3669 they occur within the XML instance.

3670 **[Rule 7-7]**

3671 Within a NIEM-conformant schema or instance, the attribute
3672 `structures:sequenceID` SHALL NOT be interpreted as meaningful beyond an
3673 indicator of sequential order of an object relative to its siblings.

3674 **[Rule 7-8]**

3675 Within a NIEM-conformant element instance, when an object O links to a
3676 metadata object via an attribute `structures:metadata`, the information in the
3677 metadata object SHALL be applied to the object O.

3678 **[Rule 7-9]**

3679 Within a NIEM-conformant element instance, when an object O1 contains an
3680 element E, with content object O2, and O2 links to a metadata object via an attribute
3681 `structures:linkMetadata`, the information in the metadata object SHALL be applied
3682 to the relationship E between O1 and O2.

3683 **[Rule 7-10]**

3684 Within a NIEM-conformant element instance, each `IDREF` contained in the value
3685 of an attribute `structures:metadata` MUST refer to an attribute `structures:id`
3686 owned by an instance of a metadata type in the same information set.

3687 **[Rule 7-11]**

3688 Within a NIEM-conformant element instance, each `IDREF` contained in the value
3689 of an attribute `structures:linkMetadata` MUST refer to an attribute
3690 `structures:id` owned by an instance of a metadata type in the same information set.

3691 **[Rule 7-12]**

3692 Within a set of NIEM-conformant element instances within an information set,
3693 any metadata element instance referred to from an element instance of some type *T*
3694 MUST be applicable to an object type *T*.

3695 **[Rule 8-0.9]**

3696 Within a NIEM-conformant schema, a complex type that is a direct extension of
3697 an XML Schema namespace simple type MAY use the same local name as the simple
3698 type, if and only if the extension adds no content other than the attribute group
3699 `structures:SimpleObjectAttributeGroup`.

3700 **[Rule 8-1]**

3701 The name of any XML Schema component defined by NIEM-conformant
3702 schemas SHALL be composed of words from the English language, using the prevalent
3703 U.S. spelling, as provided by **[OED]**.

3704 **[Rule 8-2]**

3705 The name of any XML Schema component defined by a NIEM-conformant
3706 schema SHALL contain only the following characters:

- 3707 • upper-case letters ('A'-'Z'),
- 3708 • lower-case letters ('a'-'z'),
- 3709 • digits ('0'-'9'), and
- 3710 • hyphen ('-').

3711 Other characters, such as the underscore ('_') character and the period ('.')
3712 character SHALL NOT appear in component names in NIEM-conformant schemas.

3713 **[Rule 8-3]**

3714 The hyphen character ('-') MAY appear in component names only when used as
3715 a separator between parts of a single word, phrase, or value, that would otherwise be
3716 incomprehensible without the use of a separator.

3717 **[Rule 8-4]**

3718 Within a NIEM-conformant schema, any attribute declaration SHALL have a
3719 name that begins with a lower-case letter ('a'-'z').

3720 **[Rule 8-5]**

3721 Within a NIEM-conformant schema, any XML Schema component other than an
3722 attribute declaration SHALL have a name that begins with an upper-case letter ('A'-'Z').

3723 **[Rule 8-6]**

3724 The name of any XML Schema component defined by a NIEM-conformant
3725 schema SHALL use the camel case formatting convention.

3726 **[Rule 8-7]**

3727 A NIEM-conformant schema MUST consistently use approved acronyms,
3728 abbreviations, and word truncations within defined names. The approved shortened
3729 forms are defined in Table 2: Abbreviations used in NIEM Core Names .

3730 **[Rule 8-8]**

3731 A noun used as a term in a NIEM component MUST be used in singular form,
3732 unless the concept itself is plural.

3733 **[Rule 8-9]**

3734 A verb used as a term in a NIEM component MUST be used in the present tense,
3735 unless the concept itself is past tense.

3736 **[Rule 8-10]**

3737 Articles, conjunctions and prepositions SHALL NOT be used in NIEM component
3738 names, except where they are required for clarity or by standard convention (e.g.;
3739 `PowerOfAttorneyCode`).

3740 **[Rule 8-11]**

3741 Except as specified elsewhere in this document, any element or attribute defined
3742 within a NIEM-conformant schema SHALL have a name which takes the form:

- 3743 • object class qualifier terms (0 or more)
- 3744 • an object class term (1)
- 3745 • property qualifier terms (0 or more)
- 3746 • a property term (1)
- 3747 • representation qualifier terms (0 or more)
- 3748 • a representation term (1).

3749 **[Rule 8-12]**

3750 The object class term of a NIEM component SHALL consist of a term identifying
3751 a category of concrete concepts or entities.

3752 **[Rule 8-13]**

3753 A property term SHALL describe or represent a characteristic or subpart of an
3754 entity or concept.

3755 **[Rule 8-14]**

3756 Multiple qualifier terms MAY be used within a component name as necessary to
3757 ensure clarity and uniqueness within its namespace and usage context.

3758 **[Rule 8-15]**

3759 The number of qualifier terms SHOULD be limited to the absolute minimum
3760 required to make the component name unique and understandable.

3761 **[Rule 8-16]**

3762 The order of qualifiers SHALL NOT be used to differentiate names.

3763 **[Rule 8-17]**

3764 If any word in the representation term is redundant with any word in the property
3765 term, one occurrence SHOULD be deleted.

3766 **[Rule 8-18]**

3767 Within a NIEM-conformant schema, the name of an element declaration that is of
3768 simple content MUST use a representation term found in Table 3: Representation Terms.

3769 **[Rule 8-19]**

3770 Within a NIEM-conformant schema, the name of an element declaration that is of
3771 complex content, and which corresponds to a concept listed in Table 3: Representation
3772 Terms, MUST use a representation term from that table.

3773 **[Rule 8-20]**

3774 Within a NIEM-conformant schema, the name of an element declaration which is
3775 of complex content and which does not correspond to a concept listed in Table 3:
3776 Representation Terms, MUST NOT use a representation term from that table.

3777 **[Rule 8-21]**

3778 Within a NIEM-conformant schema, the name of an attribute declaration MUST
3779 use a representation term from Table 3: Representation Terms.

3780 **[Rule 8-22]**

3781 Within a NIEM-conformant schema, the name of any type definition MUST use
3782 the representation term *Type*.

3783 **[Rule 8-23]**

3784 Within a NIEM-conformant schema, the name of any simple type definition
3785 SHALL use the representation term qualifier *Simple*. This qualifier SHALL appear after
3786 any other representation term qualifiers.

3787 **[Rule 8-24]**

3788 Within a NIEM-conformant schema, the name of any code type SHALL use the
3789 representation term qualifier *Code*.

3790 **[Rule 8-25]**

3791 Within a NIEM-conformant schema, any type definition which has a base type
3792 definition of a code type or which is transitively based on a code type SHALL have a
3793 name which uses the representation term qualifier *Code*.

3794 **[Rule 8-26]**

3795 Within a NIEM-conformant schema, any association type SHALL have a name
3796 that uses the representation term qualifier *Association*. Types other than association
3797 types SHALL NOT use the representation term qualifier *Association*.

3798 **[Rule 8-27]**

3799 Within a NIEM-conformant schema, any augmentation type SHALL have a name
3800 that uses the representation term qualifier *Augmentation*. Types other than
3801 augmentation types SHALL NOT use the representation term qualifier *Augmentation*.

3802 **[Rule 8-28]**

3803 Within a NIEM-conformant schema, any metadata type SHALL have a name that
3804 uses the representation term qualifier *Metadata*. Types other than metadata types
3805 SHALL NOT use the representation term qualifier *Metadata*.

3806 **[Rule 8-29]**

3807 Within a NIEM-conformant schema, the name of any attribute group definition
3808 schema component SHALL use the representation term *AttributeGroup*.

3809 **[Rule 8-30]**

3810 Within a NIEM-conformant schema, the name of any reference element SHALL
3811 use the representation term suffix *Reference*.

3812 **[Rule 8-31]**

3813 Within a NIEM-conformant schema, the name of an association element SHALL
3814 use the representation term qualifier *Association*.

3815 **[Rule 8-32]**

3816 Within a NIEM-conformant schema, the name of an augmentation element
3817 SHALL use the representation term *Augmentation*.

3818 **[Rule 8-33]**

3819 Within a NIEM-conformant schema, the name of a metadata element SHALL use
3820 the representation term *Metadata*.

3821 **[Rule 8-34]**

3822 Within a NIEM-conformant schema, the name of a role SHALL use the property
3823 term *RoleOf*.

3824

Appendix D. Name Syntax for Special Components

The following table summarizes NIEM general naming syntax for special components and their associated types. Refer to Sections 8.12 and 8.13 for the specific rules associated with this table.

Note this table does not mention the general syntax for standard types and properties introduced in Sections 8.12 and 8.13.

Table 4: Name Syntax for Special Components

Name Syntax *	Notes
Association	
[Property]Association	Preferred: [Property] describes relationship
[Object1][Object2]Association	Alternate 1: related objects
[Object]Association	Alternate 2: related objects are same class
Role Reference	
RoleOf[Object]Reference	Element in the role that references base type
Type Augmentation	
[Object][Property]Augmentation	[Object][Property] is from type augmented
Metadata	
[Property]Metadata	
Adapter	
[Object][Property]Adapter	
Abstract	
[Object][Property]	Preferred
[Object][Property]Abstract	Alternate: when required to prevent name clash

* Object and Property refer to **[ISO 11179 Part 5]** terms in a component name.

3834

Appendix E. Representation Terms

3835

The following table lists the standard set of representation terms for use in the representation portion of NIEM-conformant component name. Refer to Section 8.11, Representation Term, for the specific rules associated with this table.

3836

3837

3838

Primary Representation Term	Secondary Representation Term	Definition
Amount	-	A number of monetary units specified in a currency where the unit of currency is explicit or implied.
BinaryObject	-	A set of finite-length sequences of binary octets.
	Graphic	A diagram, graph, mathematical curves, or similar representation
	Picture	A visual representation of a person, object, or scene
	Sound	A representation for audio
	Video	A motion picture representation; may include audio encoded within
Code		A character string (letters, figures or symbols) that for brevity, language independence, or precision, represents a definitive value of an attribute.
DateTime		A particular point in the progression of time together with relevant supplementary information.
	Date	A particular day, month, and year in the Gregorian calendar.
	Time	A particular point in the progression of time within an unspecified 24 hour day.
ID		A character string to identify and distinguish uniquely, one instance of an object in an identification scheme from all other objects in the same scheme together with relevant supplementary information.

	URI	A string of characters used to identify (or name) a resource. The main purpose of this identifier is to enable interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols. A URI is either a Uniform Resource Locator (URL) or a Uniform Resource Name (URN). The specific syntax for each is defined by [RFC3986] .
Indicator		A list of two mutually exclusive Boolean values that express the only possible states of a property.
Measure		A numeric value determined by measuring an object along with the specified unit of measure.
Numeric		Numeric information that is assigned or is determined by calculation, counting, or sequencing. It does not require a unit of quantity or unit of measure.
	Value	A result of a calculation
	Rate	A representation of a ratio where the two units are not included.
	Percent	A representation of a ratio in which the two units are the same.
Quantity		A counted number of non-monetary units possibly including fractions.
Text	-	A character string (i.e. a finite sequence of characters) generally in the form of words of a language.
	Name	A word or phrase that constitutes the distinctive designation of a person, place, thing or concept.

3839

3840
3841
3842
3843
3844

Appendix F. Documentation Standard Opening Phrases

This listing of standard opening phrases is informative only. For reference purposes, it repeats a table that appears in Section 6.2.1, Human-Readable Documentation.

ThisComponent Class	Definition opening phrase
Abstract	"A data concept for a ..."
Association	"A relationship ..."
Augmentation	"Supplements ..."
Entities and properties of such	"A (An) ..."
Indicator	"True if ...; false otherwise/if..."
Role	"Acts as ..."
Type	"A data type for ..."
Role	"Acts as ..."

3845

3846

Appendix G. NIEM Core Abbreviations

3847

This listing of abbreviations used in NIEM Core is informative only. For reference purposes, it repeats a table that appears in Section 8.5, Use of Acronyms and Abbreviations.

3848

3849

3850

Abbreviation	Full Meaning
ANSI	American National Standards Institute
CMV	Commercial Motor Vehicle
DEA	Drug Enforcement Agency
DNA	Deoxyribonucleic Acid
FGI	Foreign Government Information
FIPS	Federal Information Processing Standard
IC	Intelligence Community
ID	Identifier
IP	Internet Protocol
ISO	International Standards Organization
LIS	NCIC code list for license state
LSTA	NCIC code list for state/country index
MCO	Manufacturer's Certificate of Origin
MGRS	Military Grid Reference System
MSRP	Manufacturer's Suggested Retail Price
NANP	North American Numbering Plan
NCIC	National Crime Information Center
NCTC	National Counter Terrorist Center
NIBRS	National Incident Based Reporting System
NLETS	The International Justice & Public Safety Information Sharing Network (formerly known as the National Law Enforcement Teletype System)
ORI	Organization Identifier (Orion)
RES	NCIC code list for registration state for boat registrations
RF	Radio Frequency
SIM	Subscriber Identity Module
SSN	Social Security Number
TYP	NCIC code list for gun type
TYPO	NCIC code list for ORI type
URI	Uniform Resource Identifier
US	United States
UTM	Universal Transverse Mercator
VIN	Vehicle Identification Number
VINA	Vehicle Identification Number Analysis

3851

3852 Appendix H. Supporting Schemas

3853 NIEM provides a set of schemas which underlie the data model schemas. These
3854 schemas do not define data model content; they don't define people, or vehicles, or
3855 relationships between them. Instead, these schemas define the fundamental framework
3856 on which the data model is built.

3857 There are two supporting schemas. The first is called `appinfo`, and is the namespace
3858 for application information that supports data model definitions. The second is called
3859 `structures`, and is the namespace for basic types that augment the mechanisms of
3860 XML Schema for more sophisticated data modeling and information exchanges.

3861 This appendix defines and discusses each of the framework components in the two
3862 supporting schemas. At the conclusion of the discussion of each schema, the full
3863 schema is provided as a reference.

3864 This appendix also includes a directory listing of all the reference schemas that are part
3865 of NIEM 2.0.

3866 The `appinfo` namespace

3867 The `appinfo` schema provides support for high level data model concepts and additional
3868 syntax to support the NIEM conceptual model and validation of NIEM-conformant
3869 instances.

3870 Schema document element

```
3871 <xsd:schema  
3872   xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
3873   xmlns:i="http://niem.gov/niem/appinfo/2.0"  
3874   xmlns:s="http://niem.gov/niem/structures/2.0"  
3875   targetNamespace="http://niem.gov/niem/appinfo/2.0"  
3876   attributeFormDefault="qualified" version="1">
```

3877 Discussion

3878 The namespace for the `appinfo` namespace is
3879 `http://niem.gov/niem/appinfo/2.0`.

3880 Element `appinfo:Resource`

```
3881 <xsd:element name="Resource">  
3882   <xsd:complexType>  
3883     <xsd:attribute name="name" type="xsd:NCName" use="required"/>  
3884   </xsd:complexType>  
3885 </xsd:element>
```

3886 Discussion

3887 The `Resource` element provides a method for application information to define a
3888 name within a schema, without the name being bound to a schema component.
3889 This is used by the `structures` schema to define names for
3890 `structures:Object` and `structures:Association`.

3891

Element appinfo:Deprecated3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902

```

<xsd:element name="Deprecated">
  <xsd:complexType>
    <xsd:attribute name="value" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:boolean">
          <xsd:pattern value="true"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>

```

3903

Discussion3904
3905
3906

The `Deprecated` element provides a method for identifying components as being deprecated. A deprecated component is one which is provided, but whose use is not recommended.

3907

Element appinfo:Base3908
3909
3910
3911
3912
3913

```

<xsd:element name="Base">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    <xsd:attribute name="namespace" type="xsd:anyURI" use="optional"/>
  </xsd:complexType>
</xsd:element>

```

3914

Discussion3915
3916
3917

The `Base` element provides a mechanism for indicating base types and base elements in schema, for the cases in which XML Schema mechanisms are insufficient. For example, it is used to indicate `Object` or `Association` bases.

3918

Element appinfo:ReferenceTarget3919
3920
3921
3922
3923
3924

```

<xsd:element name="ReferenceTarget">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    <xsd:attribute name="namespace" type="xsd:anyURI" use="optional"/>
  </xsd:complexType>
</xsd:element>

```

3925

Discussion3926
3927
3928

The `ReferenceTarget` element indicates a NIEM type which may be a target (that is, a destination) of a NIEM reference element. It may be used in combinations to indicate a set of valid types.

3929

Element appinfo:AppliesTo3930
3931
3932
3933
3934
3935

```

<xsd:element name="AppliesTo">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    <xsd:attribute name="namespace" type="xsd:anyURI" use="optional"/>
  </xsd:complexType>
</xsd:element>

```

3936 **Discussion**

3937 The `AppliesTo` element is used in two ways. First, it indicates the set of types
3938 to which a metadata type may be applied. Second, it indicates the set of types to
3939 which an augmentation element may be applied.

3940 **Element `appinfo:ConformantIndicator`**

3941

```
<xsd:element name="ConformantIndicator" type="boolean"/>
```

3942 **Discussion**

3943 The `ConformantIndicator` element may be used in two ways. First, it is
3944 included as application information for a schema document element to indicate
3945 that the schema is NIEM-conformant. Second, it is used as application
3946 information of a namespace import to indicate that the schema is not NIEM-
3947 conformant.

3948 **Element `appinfo:ExternalAdapterTypeIndicator`**

3949

```
<xsd:element name="ExternalAdapterTypeIndicator" type="boolean"/>
```

3950 **Discussion**

3951 The `ExternalAdapterTypeIndicator` element indicates that a complex type
3952 is an external adapter type. Such a type is one that is composed of elements
3953 and attributes from non-NIEM-conformant schemas. The indicator allows
3954 schema processors to switch to alternative processing modes when processing
3955 NIEM-conformant versus non-NIEM-conformant content.

3956

3957

Full XML Schema for Appinfo Namespace

3958

3959

3960

3961

3962

3963

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:i="http://niem.gov/niem/appinfo/2.0"
xmlns:s="http://niem.gov/niem/structures/2.0"
targetNamespace="http://niem.gov/niem/appinfo/2.0"
attributeFormDefault="qualified" version="1">
```

3964

3965

3966

3967

3968

```
<xsd:element name="Resource">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
  </xsd:complexType>
</xsd:element>
```

3969

3970

3971

3972

3973

3974

3975

3976

3977

3978

3979

```
<xsd:element name="Deprecated">
  <xsd:complexType>
    <xsd:attribute name="value" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:boolean">
          <xsd:pattern value="true"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>
```

3980

3981

3982

3983

3984

3985

```
<xsd:element name="Base">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    <xsd:attribute name="namespace" type="xsd:anyURI" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

3986

3987

3988

3989

3990

3991

```
<xsd:element name="ReferenceTarget">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    <xsd:attribute name="namespace" type="xsd:anyURI" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

3992

3993

3994

3995

3996

3997

```
<xsd:element name="AppliesTo">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    <xsd:attribute name="namespace" type="xsd:anyURI" use="optional"/>
  </xsd:complexType>
</xsd:element>
```

3998

3999

4000

4001

```
<xsd:element name="ConformantIndicator" type="xsd:boolean"/>
<xsd:element name="ExternalAdapterTypeIndicator" type="xsd:boolean"/>
</xsd:schema>
```

4002 The structures schema

4003 The `structures` schema provides support for fundamental NIEM linking mechanisms,
4004 as well as providing base types for definition of NIEM-conformant types.

4005 Schema document element

```
4006 <?xml version="1.0" encoding="UTF-8"?>
4007 <xsd:schema
4008   targetNamespace="http://niem.gov/niem/structures/2.0"
4009   version="1"
4010   xmlns:appinfo="http://niem.gov/niem/appinfo/2.0"
4011   xmlns:s="http://niem.gov/niem/structures/2.0"
4012   xmlns="http://www.w3.org/2001/XMLSchema">
```

4013 Discussion

4014 The target namespace for the `structures` schema is
4015 `http://niem.gov/niem/structures/2.0`.

4016 Import of `appinfo`

```
4017 <xsd:import
4018   schemaLocation="../../appinfo/2.0/appinfo.xsd"
4019   namespace="http://niem.gov/niem/appinfo/2.0"/>
```

4020 Discussion

4021 The `structures` schema uses components from the `appinfo` namespace.

4022 Resource `structures:Object`

```
4023 <xsd:annotation>
4024   <xsd:appinfo>
4025     <i:Resource i:name="Object"/>
4026   </xsd:appinfo>
4027 </xsd:annotation>
```

4028 Discussion

4029 The `Object` resource defines an identifier which acts as a conceptual base for
4030 objects in NIEM-conformant schemas.

4031 Resource `structures:Association`

```
4032 <xsd:annotation>
4033   <xsd:appinfo>
4034     <i:Resource i:name="Association"/>
4035   </xsd:appinfo>
4036 </xsd:annotation>
```

4037 Discussion

4038 The `Association` resource defines an identifier which acts as a conceptual
4039 base for association in NIEM-conformant schemas.

4040

Attribute structures:id

4041

```
<xsd:attribute name="id" type="ID"/>
```

4042

Discussion

4043

The `id` attribute is used to define XML IDs for NIEM objects. These IDs may be targets of reference elements, metadata attributes, and link metadata attributes.

4044

4045

Attribute structures:linkMetadata

4046

```
<xsd:attribute name="linkMetadata" type="IDREFS"/>
```

4047

Discussion

4048

The `linkMetadata` attribute allows an element to point to metadata that affects the relationship between the context and the value of the object.

4049

4050

Attribute structures:metadata

4051

```
<xsd:attribute name="metadata" type="IDREFS"/>
```

4052

Discussion

4053

The attribute `metadata` allows an object to point to metadata that affects itself.

4054

Attribute structures:ref

4055

```
<xsd:attribute name="ref" type="IDREF"/>
```

4056

Discussion

4057

The `ref` attribute is used by reference elements in NIEM to refer to an object via an ID reference, rather than including the object itself as element content.

4058

4059

Attribute structures:sequenceID

4060

```
<xsd:attribute name="sequenceID" type="integer"/>
```

4061

Discussion

4062

The `sequenceID` attribute allows a series of elements to define a sequence for content that does not correspond to the order of element declarations within a type. This attribute may override the sequence of elements appearing within an instance.

4063

4064

4065

4066

Attribute group structures:SimpleObjectAttributeGroup

4067

```
<xsd:attributeGroup name="SimpleObjectAttributeGroup">  
  <xsd:attribute ref="s:id"/>  
  <xsd:attribute ref="s:metadata"/>  
  <xsd:attribute ref="s:linkMetadata"/>  
</xsd:attributeGroup>
```

4068

4069

4070

4071

4072 **Discussion**

4073 The SimpleObjectAttributeGroup attribute group provides a collection of
4074 attributes which are appropriate for definition of object types.

4075 **Element structures:Augmentation**

4076 `<xsd:element name="Augmentation" type="s:AugmentationType"`
4077 `abstract="true"/>`

4078 **Discussion**

4079 The Augmentation element provides a substitution group head for
4080 augmentations. The designer of a message or object may use this element
4081 within an object definition. This will allow the selection of augmentations
4082 dynamically, at run time (or at least schema selection time) rather than at schema
4083 authoring time.

4084 **Element structures:Metadata**

4085 `<xsd:element name="Metadata" type="s:MetadataType" abstract="true"/>`

4086 **Discussion**

4087 The Metadata element provides a substitution group head for metadata. Like
4088 the substitution group head for augmentations, this allows selection of metadata
4089 to be decided late in message creation, rather than at schema authoring time.
4090 This element may also be used to provide a single point in a container where all
4091 metadata for a message may be deposited.

4092 **Complex type structures:AugmentationType**

4093 `<xsd:complexType name="AugmentationType" abstract="true">`
4094 `<xsd:attribute ref="s:id"/>`
4095 `<xsd:attribute ref="s:metadata"/>`
4096 `</xsd:complexType>`

4097 **Discussion**

4098 The AugmentationType type is a base type for all augmentations. An
4099 augmentation may have metadata and an ID, but may not have link metadata, as
4100 it does not establish a relationship between its value and its context. The
4101 individual element contents of an augmentation, however, do establish a
4102 relationship between the context of the augmentation and the values of the
4103 individual elements.

4104 **Type structures:ComplexObjectType**

4105 `<xsd:complexType name="ComplexObjectType" abstract="true">`
4106 `<xsd:attribute ref="s:id"/>`
4107 `<xsd:attribute ref="s:metadata"/>`
4108 `<xsd:attribute ref="s:linkMetadata"/>`
4109 `</xsd:complexType>`

4110 **Discussion**

4111 The `ComplexObjectType` type provides a base class for object definition,
4112 association definitions, and external adapter type definitions. An instance of one
4113 of these types may have an ID. It may have metadata as it establishes the
4114 existence of an object (maybe a conceptual object). It may also have link
4115 metadata, as an element of one of these types establishes a relationship
4116 between its value and its context.

4117 **Type structures:MetadataType**

```
4118 <xsd:complexType name="MetadataType" abstract="true">  
4119   <xsd:attribute ref="s:id"/>  
4120 </xsd:complexType>
```

4121 **Discussion**

4122 The `MetadataType` type is a base class for metadata type definition. This type
4123 provides only an ID, as the metadata may be referenced. It does not itself have
4124 metadata, and does not have link metadata.

4125 **Type structures:ReferenceType**

```
4126 <xsd:complexType name="ReferenceType" final="#all">  
4127   <xsd:attribute ref="s:id"/>  
4128   <xsd:attribute ref="s:ref"/>  
4129   <xsd:attribute ref="s:linkMetadata"/>  
4130 </xsd:complexType>
```

4131 **Discussion**

4132 The `ReferenceType` type is the type of all reference elements within NIEM-
4133 conformant schemas. This type provides a reference attribute, to reference an
4134 object defined elsewhere. It includes an ID, as the link established by a
4135 reference element may need to be identified, and it includes link metadata, as an
4136 element of this type establishes a relationship between its context and the
4137 referenced object. It does not contain metadata, as it does not itself establish the
4138 existence of an object; it relies on a definition located elsewhere.

4139

Full XML Schema for Structures Namespace

```

4140 <?xml version="1.0" encoding="UTF-8"?>
4141 <xsd:schema
4142   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4143   xmlns:i="http://niem.gov/niem/appinfo/2.0"
4144   xmlns:s="http://niem.gov/niem/structures/2.0"
4145   targetNamespace="http://niem.gov/niem/structures/2.0"
4146   version="1">
4147
4148   <xsd:import
4149     schemaLocation="../../appinfo/2.0/appinfo.xsd"
4150     namespace="http://niem.gov/niem/appinfo/2.0"/>
4151
4152   <xsd:annotation>
4153     <xsd:appinfo>
4154       <i:Resource i:name="Object"/>
4155     </xsd:appinfo>
4156   </xsd:annotation>
4157
4158   <xsd:annotation>
4159     <xsd:appinfo>
4160       <i:Resource i:name="Association"/>
4161     </xsd:appinfo>
4162   </xsd:annotation>
4163
4164   <xsd:attribute name="id" type="xsd:ID"/>
4165   <xsd:attribute name="linkMetadata" type="xsd:IDREFS"/>
4166   <xsd:attribute name="metadata" type="xsd:IDREFS"/>
4167   <xsd:attribute name="ref" type="xsd:IDREF"/>
4168   <xsd:attribute name="sequenceID" type="xsd:integer"/>
4169
4170   <xsd:attributeGroup name="SimpleObjectAttributeGroup">
4171     <xsd:attribute ref="s:id"/>
4172     <xsd:attribute ref="s:metadata"/>
4173     <xsd:attribute ref="s:linkMetadata"/>
4174   </xsd:attributeGroup>
4175
4176   <xsd:element name="Augmentation" type="s:AugmentationType"
4177     abstract="true"/>
4178   <xsd:element name="Metadata" type="s:MetadataType" abstract="true"/>
4179
4180   <xsd:complexType name="AugmentationType" abstract="true">
4181     <xsd:attribute ref="s:id"/>
4182     <xsd:attribute ref="s:metadata"/>
4183     <xsd:attribute ref="s:linkMetadata"/>
4184   </xsd:complexType>
4185
4186   <xsd:complexType name="ComplexObjectType" abstract="true">
4187     <xsd:attribute ref="s:id"/>
4188     <xsd:attribute ref="s:metadata"/>
4189     <xsd:attribute ref="s:linkMetadata"/>
4190   </xsd:complexType>
4191
4192   <xsd:complexType name="MetadataType" abstract="true">
4193     <xsd:attribute ref="s:id"/>
4194   </xsd:complexType>
4195
4196   <xsd:complexType name="ReferenceType" final="#all">
4197     <xsd:attribute ref="s:id"/>
4198     <xsd:attribute ref="s:ref"/>
4199     <xsd:attribute ref="s:linkMetadata"/>
4200   </xsd:complexType>
4201
4202 </xsd:schema>

```

4196

NIEM 2.0 Reference Schemas – Directory Listing

```
4197 niem
4198 |
4199 |
4200 |---ansi-nist
4201 |   |---2.0
4202 |       ansi-nist.xsd
4203 |
4204 |---ansi_d20
4205 |   |---2.0
4206 |       ansi_d20.xsd
4207 |
4208 |---apco
4209 |   |---2.0
4210 |       apco.xsd
4211 |
4212 |---appinfo
4213 |   |---2.0
4214 |       appinfo.xsd
4215 |
4216 |---atf
4217 |   |---2.0
4218 |       atf.xsd
4219 |
4220 |---census
4221 |   |---2.0
4222 |       census.xsd
4223 |
4224 |---dea
4225 |   |---2.0
4226 |       dea.xsd
4227 |
4228 |---dod_jcs-pub2.0-misc
4229 |   |---2.0
4230 |       dod_jcs-pub2.0-misc.xsd
4231 |
4232 |---domains
4233 |   |---emergencyManagement
4234 |       |---2.0
4235 |           emergencyManagement.xsd
4236 |
4237 |   |---immigration
4238 |       |---2.0
4239 |           immigration.xsd
4240 |
4241 |   |---infrastructureProtection
4242 |       |---2.0
4243 |           infrastructureProtection.xsd
4244 |
4245 |   |---intelligence
4246 |       |---2.0
4247 |           intelligence.xsd
4248 |
4249 |   |---internationalTrade
4250 |       |---2.0
4251 |           internationalTrade.xsd
```

```

4252 |
4253 |   |
4254 |   |   |
4255 |   |   |   |
4256 |   |   |   |   |
4257 |   |   |   |   |   |
4258 |   |   |   |   |   |   |
4259 |   |   |   |   |   |   |   |
4260 |   |   |   |   |   |   |   |
4261 |   |   |   |   |   |   |   |
4262 |   |   |   |   |   |   |   |   |
4263 |   |   |   |   |   |   |   |   |
4264 |   |   |   |   |   |   |   |   |
4265 |   |   |   |   |   |   |   |   |
4266 |   |   |   |   |   |   |   |   |
4267 |   |   |   |   |   |   |   |   |
4268 |   |   |   |   |   |   |   |   |
4269 |   |   |   |   |   |   |   |   |
4270 |   |   |   |   |   |   |   |   |
4271 |   |   |   |   |   |   |   |   |
4272 |   |   |   |   |   |   |   |   |
4273 |   |   |   |   |   |   |   |   |
4274 |   |   |   |   |   |   |   |   |
4275 |   |   |   |   |   |   |   |   |
4276 |   |   |   |   |   |   |   |   |
4277 |   |   |   |   |   |   |   |   |
4278 |   |   |   |   |   |   |   |   |
4279 |   |   |   |   |   |   |   |   |
4280 |   |   |   |   |   |   |   |   |
4281 |   |   |   |   |   |   |   |   |
4282 |   |   |   |   |   |   |   |   |
4283 |   |   |   |   |   |   |   |   |
4284 |   |   |   |   |   |   |   |   |
4285 |   |   |   |   |   |   |   |   |
4286 |   |   |   |   |   |   |   |   |
4287 |   |   |   |   |   |   |   |   |
4288 |   |   |   |   |   |   |   |   |
4289 |   |   |   |   |   |   |   |   |
4290 |   |   |   |   |   |   |   |   |
4291 |   |   |   |   |   |   |   |   |
4292 |   |   |   |   |   |   |   |   |
4293 |   |   |   |   |   |   |   |   |
4294 |   |   |   |   |   |   |   |   |
4295 |   |   |   |   |   |   |   |   |
4296 |   |   |   |   |   |   |   |   |
4297 |   |   |   |   |   |   |   |   |
4298 |   |   |   |   |   |   |   |   |
4299 |   |   |   |   |   |   |   |   |
4300 |   |   |   |   |   |   |   |   |
4301 |   |   |   |   |   |   |   |   |
4302 |   |   |   |   |   |   |   |   |
4303 |   |   |   |   |   |   |   |   |
4304 |   |   |   |   |   |   |   |   |
4305 |   |   |   |   |   |   |   |   |
4306 |   |   |   |   |   |   |   |   |
4307 |   |   |   |   |   |   |   |   |
4308 |   |   |   |   |   |   |   |   |

```

jxdm
 4.0
 jxdm.xsd
 screening
 2.0
 screening.xsd
 edxl
 2.0
 edxl.xsd
 edxl-cap
 2.0
 edxl-cap.xsd
 edxl-de
 2.0
 edxl-de.xsd
 external
 cap
 1.1
 cap.xsd
 de
 1.0
 de.xsd
 dhs-gmo
 AS
 mobileObject
 1.0.0
 mobileObject.xsd
 multiModalRoute
 1.0.0
 multiModalRoute.xsd
 iai-ifc
 rc2
 dhs-gmo
 1.0.0
 IFC2X2_FINAL.xsd
 iso-10303-step
 2
 dhs-gmo
 1.0.0
 configuration.xsd
 ex.xsd
 iso-19139-gmd
 draft-0.1
 gco
 dhs-gmo

4309			1.0.0
4310			basicTypes.xsd
4311			gco.xsd
4312			gcoBase.xsd
4313			
4314		gmd	
4315		dhs-gmo	
4316		1.0.0	
4317			applicationSchema.xsd
4318			citation.xsd
4319			constraints.xsd
4320			content.xsd
4321			dataQuality.xsd
4322			distribution.xsd
4323			extent.xsd
4324			freeText.xsd
4325			gmd.xsd
4326			identification.xsd
4327			maintenance.xsd
4328			metadataApplication.xsd
4329			metadataEntity.xsd
4330			metadataExtension.xsd
4331			portrayalCatalogue.xsd
4332			referenceSystem.xsd
4333			spatialRepresentation.xsd
4334			
4335		gmx	
4336		dhs-gmo	
4337		1.0.0	
4338			catalogues.xsd
4339			odelistItem.xsd
4340			crsItem.xsd
4341			extendedTypes.xsd
4342			gmx.xsd
4343			gmxUsage.xsd
4344			uomItem.xsd
4345			
4346		gsr	
4347		dhs-gmo	
4348		1.0.0	
4349			gsr.xsd
4350			spatialReferencing.xsd
4351			
4352		gss	
4353		dhs-gmo	
4354		1.0.0	
4355			geometry.xsd
4356			gss.xsd
4357			
4358		gts	
4359		dhs-gmo	
4360		1.0.0	
4361			gts.xsd
4362			temporalObjects.xsd
4363			
4364		landxml	
4365		1.1	

```

4366      LandXML-1.1.xsd
4367
4368      |
4369      |   ogc-context
4370      |   |   1.1.0
4371      |   |   |   dhs-gmo
4372      |   |   |   |   1.0.0
4373      |   |   |   |   |   context.xsd
4374      |   |
4375      |   |   ogc-filter
4376      |   |   |   1.1.0
4377      |   |   |   |   dhs-gmo
4378      |   |   |   |   |   1.0.0
4379      |   |   |   |   |   filter.xsd
4380      |   |
4381      |   |   ogc-gml
4382      |   |   |   3.1.1
4383      |   |   |   |   dhs-gmo
4384      |   |   |   |   |   1.0.0
4385      |   |   |   |   |   gml.xsd
4386      |   |
4387      |   |   ogc-observation
4388      |   |   |   draft-0.14.5
4389      |   |   |   |   om
4390      |   |   |   |   |   dhs-gmo
4391      |   |   |   |   |   |   1.0.0
4392      |   |   |   |   |   |   commonObservation.xsd
4393      |   |   |   |   |   |   event.xsd
4394      |   |   |   |   |   |   observation.xsd
4395      |   |   |   |   |   |   observationSpecializations.xsd
4396      |   |   |   |   |   |   om.xsd
4397      |   |   |   |   |   |   procedure.xsd
4398      |   |   |   |   |   |   procedureSpecializations.xsd
4399      |   |   |   |   |
4400      |   |   |   |   |   st
4401      |   |   |   |   |   |   dhs-gmo
4402      |   |   |   |   |   |   |   1.0.0
4403      |   |   |   |   |   |   |   simpleTypeDerivation.xsd
4404      |   |   |   |   |   |
4405      |   |   |   |   |   |   swe
4406      |   |   |   |   |   |   |   dhs-gmo
4407      |   |   |   |   |   |   |   |   1.0.0
4408      |   |   |   |   |   |   |   |   discreteCoverage.xsd
4409      |   |   |   |   |   |   |   |   phenomenon.xsd
4410      |   |   |   |   |   |   |   |   record.xsd
4411      |   |   |   |   |   |   |   |   recordType.xsd
4412      |   |   |   |   |   |   |   |   swe.xsd
4413      |   |   |   |   |   |   |   |   SWE_basicTypes.xsd
4414      |   |   |   |   |   |   |   |   temporalAggregates.xsd
4415      |   |
4416      |   |   ogc-openls
4417      |   |   |   1.1.0
4418      |   |   |   |   dhs-gmo
4419      |   |   |   |   |   1.0.0
4420      |   |   |   |   |   |   ols.xsd
4421      |   |
4422      |   |   ogc-ows
4423      |   |   |   1.0.0

```

4423		└─dhs-gmo	
4424		└─1.0.0	
4425			ows.xsd
4426			
4427	└─ogc-sld		
4428	└─1.0.20		
4429	└─dhs-gmo		
4430	└─1.0.0		
4431			sld.xsd
4432			
4433	└─ogc-swe-common		
4434	└─1.0.0		
4435	└─dhs-gmo		
4436	└─1.0.0		
4437			data.xsd
4438			parameters.xsd
4439			positionData.xsd
4440			sweCommon.xsd
4441			
4442	└─ogc-wfs		
4443	└─1.1.0		
4444	└─dhs-gmo		
4445	└─1.0.0		
4446			wfs.xsd
4447			
4448	└─urisa-street-address		
4449	└─draft-0.2.0		
4450	└─dhs-gmo		
4451	└─1.0.0		
4452			StreetAddressDataStandard.xsd
4453			
4454	└─w3c-xlink		
4455	└─1.0		
4456	└─dhs-gmo		
4457	└─1.0.0		
4458			xlinks.xsd
4459			
4460	└─w3c-xml		
4461	└─1998		
4462			xml.xsd
4463			
4464	└─fbi		
4465	└─2.0		
4466			fbi.xsd
4467			
4468	└─fips_10-4		
4469	└─2.0		
4470			fips_10-4.xsd
4471			
4472	└─fips_5-2		
4473	└─2.0		
4474			fips_5-2.xsd
4475			
4476	└─fips_6-4		
4477	└─2.0		
4478			fips_6-4.xsd
4479			

```
4480 |-----geospatial
4481 |       |-----2.0
4482 |               geospatial.xsd
4483 |
4484 |-----have
4485 |       |-----2.0
4486 |               have.xsd
4487 |
4488 |-----hazmat
4489 |       |-----2.0
4490 |               hazmat.xsd
4491 |
4492 |-----iso_3166
4493 |       |-----2.0
4494 |               iso_3166.xsd
4495 |
4496 |-----iso_4217
4497 |       |-----2.0
4498 |               iso_4217.xsd
4499 |
4500 |-----iso_639-3
4501 |       |-----2.0
4502 |               iso_639-3.xsd
4503 |
4504 |-----itis
4505 |       |-----2.0
4506 |               itis.xsd
4507 |
4508 |-----lasd
4509 |       |-----2.0
4510 |               lasd.xsd
4511 |
4512 |-----mmucc_2
4513 |       |-----2.0
4514 |               mmucc_2.xsd
4515 |
4516 |-----mn_offense
4517 |       |-----2.0
4518 |               mn_offense.xsd
4519 |
4520 |-----nga
4521 |       |-----2.0
4522 |               nga.xsd
4523 |
4524 |-----niem-core
4525 |       |-----2.0
4526 |               niem-core.xsd
4527 |
4528 |-----nlets
4529 |       |-----2.0
4530 |               nlets.xsd
4531 |
4532 |-----nonauthoritative-code
4533 |       |-----2.0
4534 |               nonauthoritative-code.xsd
4535 |
4536 |-----post-canada
```

```

4537      └── 2.0
4538          post-canada.xsd
4539
4540      └── proxy
4541          └── xsd
4542              └── 2.0
4543                  xsd.xsd
4544
4545      └── sar
4546          └── 2.0
4547              sar.xsd
4548
4549      └── structures
4550          └── 2.0
4551              structures.xsd
4552
4553      └── twpdes
4554          └── 2.0
4555              twpdes.xsd
4556
4557      └── ucr
4558          └── 2.0
4559              ucr.xsd
4560
4561      └── unece_rec20-misc
4562          └── 2.0
4563              unece_rec20-misc.xsd
4564
4565      └── usps_states
4566          └── 2.0
4567              usps_states.xsd
4568
4569      └── ut_offender-tracking-misc
4570          └── 2.0
4571              ut_offender-tracking-misc.xsd
4572
4573
4574

```


4575 Appendix I. References

- 4576 **[ARCH]:** The NIEM Reference Architecture. Not yet available.
- 4577 **[CRM]:** The Federal Enterprise Architecture Consolidated Reference Model. Available
4578 from
4579 [http://www.whitehouse.gov/omb/egov/documents/FEA_CRM_v21_Final](http://www.whitehouse.gov/omb/egov/documents/FEA_CRM_v21_Final_Dec_2006.pdf)
4580 [_Dec_2006.pdf](http://www.whitehouse.gov/omb/egov/documents/FEA_CRM_v21_Final_Dec_2006.pdf)
- 4581 **[IEPD]:** Requirements for a National Information Exchange Model (NIEM) Information
4582 Exchange Package Documentation (IEPD) Specification, Version 2.1, June 2006.
4583 Available from
4584 http://www.niem.gov/files/NIEM_IEPD_Requirements_v2_1.txt
- 4585 **[ISO 11179 Part 4]:** ISO/IEC 11179-4:2004, Information technology -- Metadata registries
4586 (MDR) -- Part 4: Formulation of data definitions. Available from
4587 [http://standards.iso.org/ittf/PubliclyAvailableStandards/c0353](http://standards.iso.org/ittf/PubliclyAvailableStandards/c035346_ISO_IEC_11179-4_2004(E).zip)
4588 [46_ISO_IEC_11179-4_2004\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c035346_ISO_IEC_11179-4_2004(E).zip)
- 4589 **[ISO 11179 Part 5]:** ISO/IEC 11179-5:2005, Information technology -- Metadata registries
4590 (MDR) -- Part 5: Naming and identification principles. Available from
4591 [http://standards.iso.org/ittf/PubliclyAvailableStandards/c0353](http://standards.iso.org/ittf/PubliclyAvailableStandards/c035347_ISO_IEC_11179-5_2005(E).zip)
4592 [47_ISO_IEC_11179-5_2005\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c035347_ISO_IEC_11179-5_2005(E).zip)
- 4593 **[OED]:** Oxford English Dictionary, Second Edition, 1989. Available from
4594 <http://dictionary.oed.com/>
- 4595 **[OJP]:** OJP Information Technology Website. Available from
4596 <http://www.it.ojp.gov/jxdm>.
- 4597 **[RDFConcepts]:** Resource Description Framework (RDF): Concepts and Abstract
4598 Syntax, W3C Recommendation 10 February 2004. Available from
4599 <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>
- 4600 RDF data model is described at [#section-data-model](#)
- 4601 **[RFC2119]:** Bradner, S. Key words for use in RFCs to Indicate Requirement Levels,
4602 IETF RFC 2119, March 1997. Available from
4603 <http://www.ietf.org/rfc/rfc2119.txt>
- 4604 **[RFC3986]:** Berners-Lee, T., et al: Uniform Resource Identifier (URI): Generic Syntax,
4605 Request for Comments 3986, January 2005. Available from
4606 <http://www.ietf.org/rfc/rfc3986.txt>
- 4607 **[SchemaForXMLSchema]:** XML Schema schema for XML Schemas: Part 1: Structures.
4608 Available from <http://www.w3.org/2001/XMLSchema.xsd>
- 4609 **[SchemaforXMLSchemaInstance]:** XML Schema instance namespace. Available from
4610 <http://www.w3.org/2001/XMLSchema-instance.xsd>
- 4611 **[XML]:** Extensible Markup Language (XML) 1.0 (Fourth Edition), W3C Recommendation
4612 16 August 2006. Available from [http://www.w3.org/TR/2006/REC-xml-](http://www.w3.org/TR/2006/REC-xml-20060816/)
4613 [20060816/](http://www.w3.org/TR/2006/REC-xml-20060816/)
- 4614 EBNF notation is described at [#sec-notation](#).
- 4615 IDREF constraint is described at [#idref](#)
- 4616 **[XML-ID]:** xml:id Version 1.0, W3C Proposed Recommendation 12 July 2005. Available
4617 from <http://www.w3.org/TR/2005/PR-xml-id-20050712/>.
- 4618 **[XMLInfoSet]:** XML Information Set (Second Edition), W3C Recommendation 4
4619 February 2004. Available from [http://www.w3.org/TR/2004/REC-xml-](http://www.w3.org/TR/2004/REC-xml-info-20040204/)
4620 [info-20040204/](http://www.w3.org/TR/2004/REC-xml-info-20040204/)

4621 **[XMLNamespaces]:** Namespaces in XML, World Wide Web Consortium 16 August
4622 2006. Available from <http://www.w3.org/TR/2006/REC-xml-names->
4623 20060816.
4624 NCName is described at #NT-NCName
4625 **[XMLNamespacesErrata]:** Namespaces in XML Errata, 6 December 2002. Available
4626 from <http://www.w3.org/XML/xml-names-19990114-errata>
4627 **[XMLSchemaDatatypes]:** XML Schema Part 2: Datatypes Second Edition, W3C
4628 Recommendation 28 October 2004. Available at
4629 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>
4630 **[XMLSchemaStructures]:** XML Schema Part 1: Structures Second Edition, W3C
4631 Recommendation 28 October 2004. Available from
4632 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>

4633 Appendix J. Glossary

4634 This glossary is informative only. It collects together all the definitions which appear in
4635 the preceding document, for the benefit of those reading a hardcopy of this document.

4636 **adapter type**

4637 An **adapter type** is a NIEM-conformant type that adapts external components for use
4638 within NIEM. An adapter type creates a new class of object that embodies a single
4639 concept composed of external components. An adapter type is defined by a NIEM-
4640 conformant schema.

4641 **application information**

4642 A component is said to have **application information** of some element **E** when the root
4643 element that defines the component has an immediate child element `xsd:annotation`,
4644 which has an immediate child element `xsd:appinfo`, which has as an immediate child
4645 the element **E**.

4646 **appinfo namespace**

4647 The **appinfo namespace** is the namespace represented by the URI
4648 "<http://niem.gov/niem/appinfo/2.0>".

4649 **association**

4650 In a NIEM-conformant schema, an **association** is an element whose type is a
4651 association type.

4652 **association type**

4653 In a NIEM-conformant schema, an **association type** is a type which establishes a
4654 relationship between objects, along with the properties of that relationship. An
4655 association type provides a structure which does not establish existence of an object, but
4656 instead specifies relationships between objects.

4657 **augmentation**

4658 An **augmentation** of a NIEM-conformant object type is a block of additional data added
4659 to an object type, in order to carry additional data beyond that of the original object
4660 definition.

4661 **augmentation type**

4662 An **augmentation type** is a complex type which provides a reusable block of data which
4663 may be added to object types or association types.

4664 **code type**

4665 A **code type** is a simple type schema component definition which contains multiple
4666 `xsd:enumeration` facets.

4667 **definition**

4668 The **definition** of a documented component is the content of the occurrence of an
4669 element `xsd:documentation` that is an immediate child of the occurrence of an
4670 element `xsd:annotation`. That element `xsd:annotation` is itself an immediate
4671 child of the element that defines the component.

4672 **deprecated component**

4673 In a particular NIEM-conformant namespace, a **deprecated component** is one whose
4674 use is not recommended, yet which is maintained in the schema for compatibility with
4675 previous versions of the namespace.

4676 **documented component**

4677 In a NIEM-conformant schema, a **documented component** is an XML Schema
4678 component that is required to have associated documentation. These schema
4679 components are required to have a textual definition for the component to be well-
4680 understood. Schemas that do not document their components accordingly are not NIEM-
4681 conformant.

4682 **external schema**

4683 An **external schema** is any non-supporting schema that is not NIEM-conformant.

4684 **metadata element**

4685 Within a NIEM-conformant schema, a **metadata element** is an element whose type is a
4686 metadata type. There are specific limitations on the meaning of a metadata element in
4687 an instance; it does not establish existence of an object, nor is it a property of its
4688 containing object.

4689 **metadata type**

4690 A **metadata type** describes data about data, that is, information which is not descriptive
4691 of objects and their relationships, but is descriptive of the data itself. It is useful to
4692 provide a general mechanism for data about data. This provides required flexibility to
4693 precisely represent information.

4694 **NIEM-conformant document**

4695 A **NIEM-conformant document** is an XML information set whose document element is
4696 defined by a NIEM-conformant schema, and which follows the rules for conformant
4697 element information items as specified by this document.

4698 **NIEM-conformant element instance**

4699 A **NIEM-conformant element instance** is an XML information item which is defined by a
4700 NIEM-conformant schema, and which follows the rules for conformant instance data as
4701 specified by this document.

4702 **NIEM-conformant schema**

4703 A **NIEM-conformant schema** is an XML document which follows the rules for NIEM-
4704 conformant schemas, as provided by this document. Any schema that follows all of the
4705 rules may be called NIEM-conformant.

4706 **object type**

4707 In a NIEM-conformant schema, an **object type** is a complex type definition, an instance
4708 of which asserts the existence of an object. An object type represents some kind of
4709 object: a thing with its own lifespan that has some existence. The object may or may not
4710 be a physical object. It may be a conceptual object.

4711 **reference element**

4712 A **reference element** is an element that refers to its value by a reference attribute,
4713 instead of carrying it as content.

4714 **RoleOf element**

4715 In a NIEM-conformant schema, a **RoleOf element** is a reference element whose type is
4716 the base type of the role.

4717 **role type**

4718 A **role type** is a type that represents a particular function, purpose, usage, or role of an
4719 object.

4720 **structures namespace**

4721 The **structures namespace** is the namespace represented by the URI
4722 "http://niem.gov/niem/structures/2.0".

4723

4724

4725 **Appendix K. Notices**

4726 This document and the information contained herein is provided on an "AS IS" basis and
4727 the authors DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
4728 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN
4729 WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF
4730 MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.